The following text is translated by the free version of Deepl (https://www.deepl.com/translator) from the original German version (https://www.prof-horst-guenther.de/mbot), as of 12. 8. 2021.

The result is 99% satisfactory. However, there are some problems with external and internal links as well as with the fonts.
Since the translated file is not editable in the free version, I could not correct this.

1. prehistory

Last year I asked my wife for an mBot for Christmas - of course under the pretext that it was for our nine-year-old grandson. And then I got it. Unfortunately, things with the grandson didn't work out quite as I had hoped. He likes to play with it, but he can't even think of programming it himself - despite Scratch. The age rating of eight years given by many suppliers should therefore be regarded with caution. Makeblock itself recommends on its packaging: "from 14". That seems more realistic to me.

Why do I want a robot like mBot in the first place?

A few years ago I bought a LEGO® Mindstorms 1.5 robot and more recently LEGO Boost. But I wasn't really happy with them. The choice of actuators and sensors was too limited for me and I didn't like the graphical programming with "puzzle pieces", especially since there was no documentation for it. I also found the mechanical stability of the models unsatisfactory.

So a new attempt with the highly praised mBot® from Makeblock.

Another grandfather had a very similar approach for his granddaughter, but several years ago. His very detailed experiences can be found in a funny PDF file worth reading: http://static.education.makeblock.com/mbotandme.pdf

After I had almost finished my field report in mid-April, I was surprised to hear that there was a successor to the mBot version 1.1 (mBot 2 (Neo)). I ordered it immediately and was able to buy it from the company Technik-LPE (https://shop.technik-lpe.de/sets/1778-makeblock-mbot-2.html).

It was also delivered promptly and caused some surprises for me. More details

shortly in a separate section: ⇒mBot2 (Neo)

2. first steps with the mBot

The assembly was no problem at all. Even my grandson could have done it without any problems.

I really like the fact that the robot can be played with immediately after assembly without any further preparations.

More details ⇒here.

3. the mBot as a robot

Of course, it is obvious to first programme the robot as it is intended by the manufacturer. Since I am like Seneca ("The path is long by teaching, short and effective by example"), I searched the Internet for appropriate sketches. I was quite successful in this. So, as far as possible, I downloaded the ready-made programmes and tried them out. Mostly, I extended them by using the LED display. However, I found it very difficult to read and went in search of an alternative. Very quickly I found the TFT display. However, this was not available from almost all suppliers. I finally found it at Generation

Robots in Luxembourg https://www.generationrobots.com/de/. However, it was more expensive there than at other shops.

The shipment then took over a week, but everything went smoothly.

3.1 Remote control

Prerequisite: mBot, LED or TFT display

A widely used application for the mBot is control via the remote control. This is also part of the standard programme, which can be downloaded via the following link

https://forum.makeblock.com/t/source-code-for-the-default-program/1526/16

(Scroll to the end of the conversation)

My grandson, born in 2011, immediately found it "cool". However, his interest quickly waned.

I have therefore created some programme variations that make it a little more challenging:

First of all, I took up a suggestion by Wolfgang Pfeffer and added the possibility to

switch on and off the blue light and siren. ⇒Fire brigade(LEGO ®) I also tried out how

well LEGO® can be combined with the mBot.

In the next variant, the mBot avoids touching: ⇒Follow hand

I also find a reaction test to sound or light signals interesting (similar to the colour play

in Bartmann (Das mBot-Buch p. 341 ff) ⇒Reaction Test

Finally, another variant was created: running through a labyrinth. The aim is to avoid touching black lines by manual control: ⇒Manual lab

3.2 Fire brigade (LEGO ®)

Prerequisite: mBot

A very interesting extension of the remote control script from the standard application comes from Wolfgang Pfeffer of the University of Passau: physical_computing_-_mbot_web_v1.0.pdf

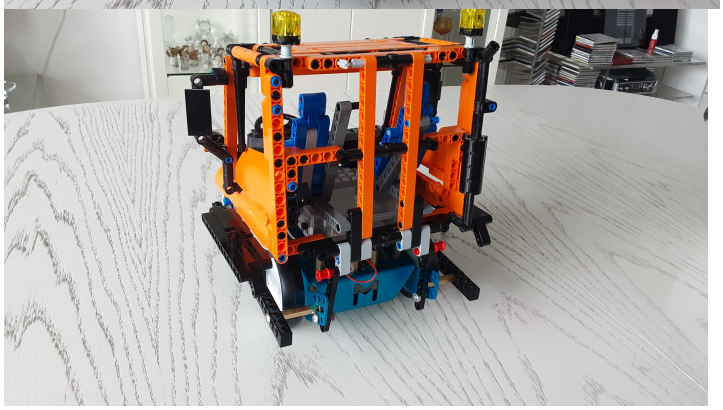
He has added to the programme the possibility to switch on and off a fire brigade sound (key B) as well as a blue light (key A).

Since even good things can be improved, I have adapted and further improved the programme. The fire brigade sound is a little more realistic and the blue light now flashes. I have also integrated the speed control from the standard programme: Different speed levels can be set by pressing the buttons 1, 2 and 3. Since the IR remote control is conceptually not completely reliable, I have included a confirmation tone. This makes it easy to understand that pressing the button to change the speed is only recognised when the mBot is not busy making the siren sound.

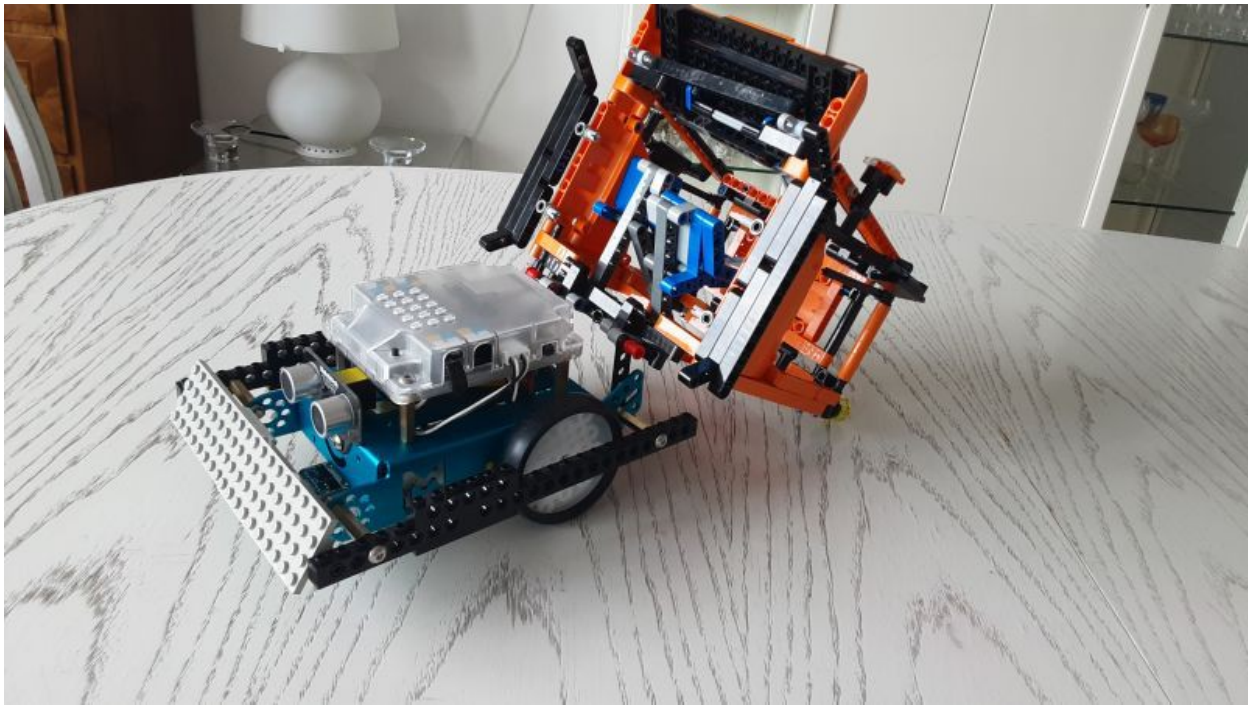
When switching the blue light and the siren, I used a little programmer's trick.

The script can be downloaded here: Fire brigade.sb2 (76 KB)

To try out how well the mBot can be combined with LEGO ®, I have slightly scaled down a largely finished model.



To enable the programme to be charged via the USB cable, the superstructure is foldable:



Overall, the construction is a bit rickety, but it serves its purpose.

Here is the YouTube video: https://youtu.be/HUo34dog5oo. I was very worried that the IR remote control would not work because the sensor is covered. Fortunately, this is not the case. However, it is easy to see how unreliably the remote control works.

See ⇒key inputwith the IR remote control.

3.3 Follow hand

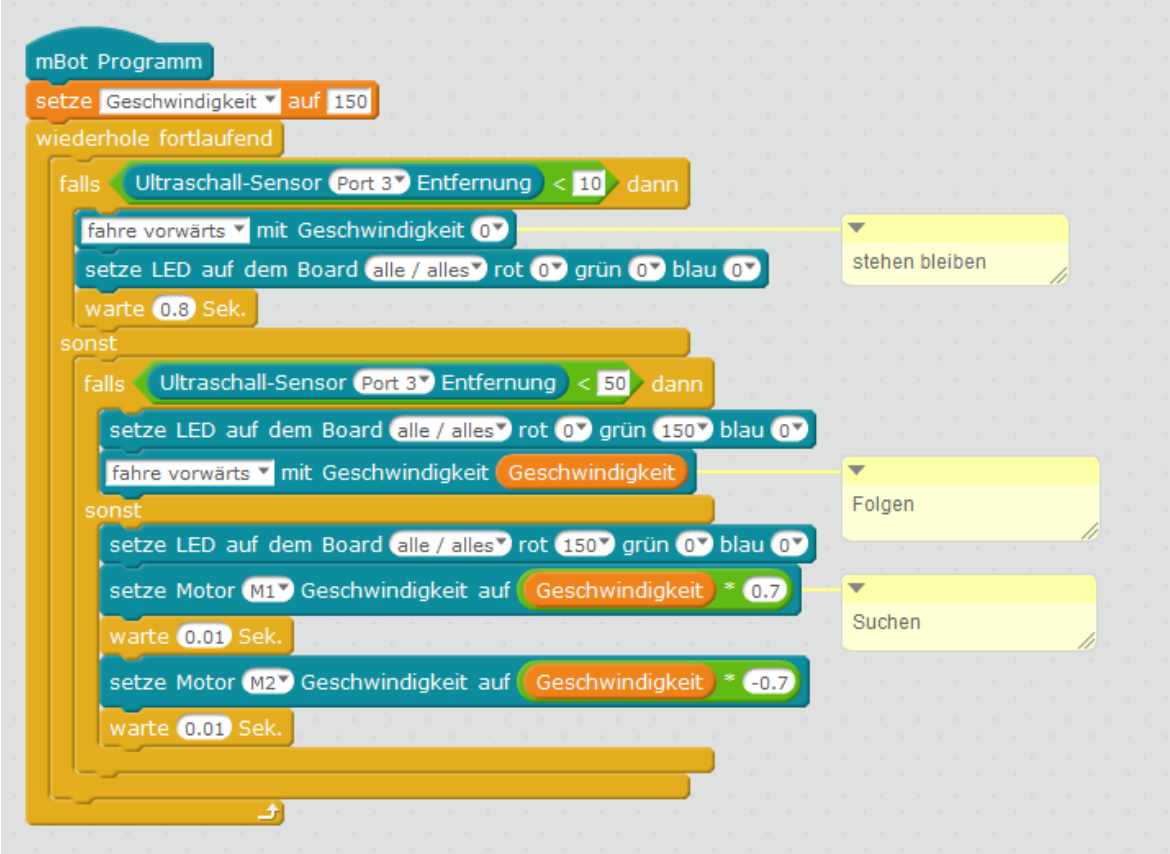
Prerequisite: mBot

With an add-on package, the mBot can be transformed into a light-following robot.

Since I do not (yet) own this additional package, I have created a simpler variant:

The mBot turns in a circle until it detects a nearby object, e.g. a hand. Then it moves towards this object. Shortly before it reaches it, it stops. If the object moves away from it, but remains within the range detected by the distance sensor, it moves forward towards it again. If the object moves away from it at a slow speed, it follows it.

Here is the YouTube video: https://youtu.be/6L25NbqV2bs



And here is the programme for download: Search_Hand.sb2 (76.1 KB)

3.4 Reaction test

Prerequisite: mBot, TFT display

When delivered, the mBot contains the so-called standard programme with its three modes:

- **Remote control with the infrared remote control**
- **Collision avoidance**
- **Track line**

As good as I think it is that you can start playing immediately without any further preparation, it is boring in the long run just to watch the robot at work.

Therefore, I have expanded each of these.

I have extended the simple line following programme so that it can be used as a reaction test.

The robot follows the black line. Randomly controlled, it generates a light and sound signal. In normal operation, the LEDs on the board light up green. When a sound is heard, the display changes to red.

This is the request to press the A button on the remote control. This event is counted. It is also measured how much time elapses between the signal and the key press.

After pressing the button, the total time, the number of measurements and the average reaction time are displayed.

A possible extension would be that, in addition to the signal, it is randomly specified which button must be pressed.

Of course, the robot can also be operated stationary. Then simply remove the *driving block.*

The video can be viewed at: https://youtu.be/0wm-jCh2iy0

The corresponding script can be downloaded here: Reaction Test.sb2 (77.2 KB)

3.5 Labyrinth (automatic)

Prerequisite: mBot, possibly LED or TFT display

There are plenty of sketches for the labyrinth on the internet. As one of the first projects, I leaned very heavily on the existing solutions. I added a special start and a special end processing.

The LED screen also shows what the mBot is currently doing.

Here is the YouTube video: Youtube video: Labyrinth as well as the sb2 file for download: Labyrinth.sb2

To put the labyrinth on paper, I proceeded as follows.

I entered *labyrinth for printing on* Google. The labyrinth can then easily (context menu via the right mouse button) be saved in a local file and prepared for printing via a suitable programme. I had good experiences with PosteRazor

(https://posterazor.de.uptodown.com/windows). However, the lines had to be repainted with a sharpie so that they could be reliably recognised by the mBot's line-following sensor.

3.6 Labyrinth (manual)

Although it is interesting to follow how the robot manages to get through the labyrinth, the game effect no longer animates at the latest after the third successful run.

The manual version offers a much longer-lasting attraction. Here, the robot is controlled using the arrow keys. The speed can also be set via the numeric keypad. The programme recognises when one of the black lines is touched or crossed via the line sequence sensor.

This is recorded with an internal counter and shown on the TFT display.

Furthermore, the time from the start of the rally is measured and displayed by pressing the A button. A running display of the time eats up too much computing time.

Here is the YouTube video: Manual Maze. The mBot sketch can be downloaded from here: Labyrinth_TFT.sb2 (79.7 KB).

3.7 Crash avoidance

Prerequisite: mBot, one or two additional line following sensors. two angles, possibly TFT display

A particularly interesting programme is crash avoidance together with obstacle detection.

There are videos on the internet and the corresponding programme for each. The first one uses two line-tracing sensors.
Avoiding crashes at the edge of the table

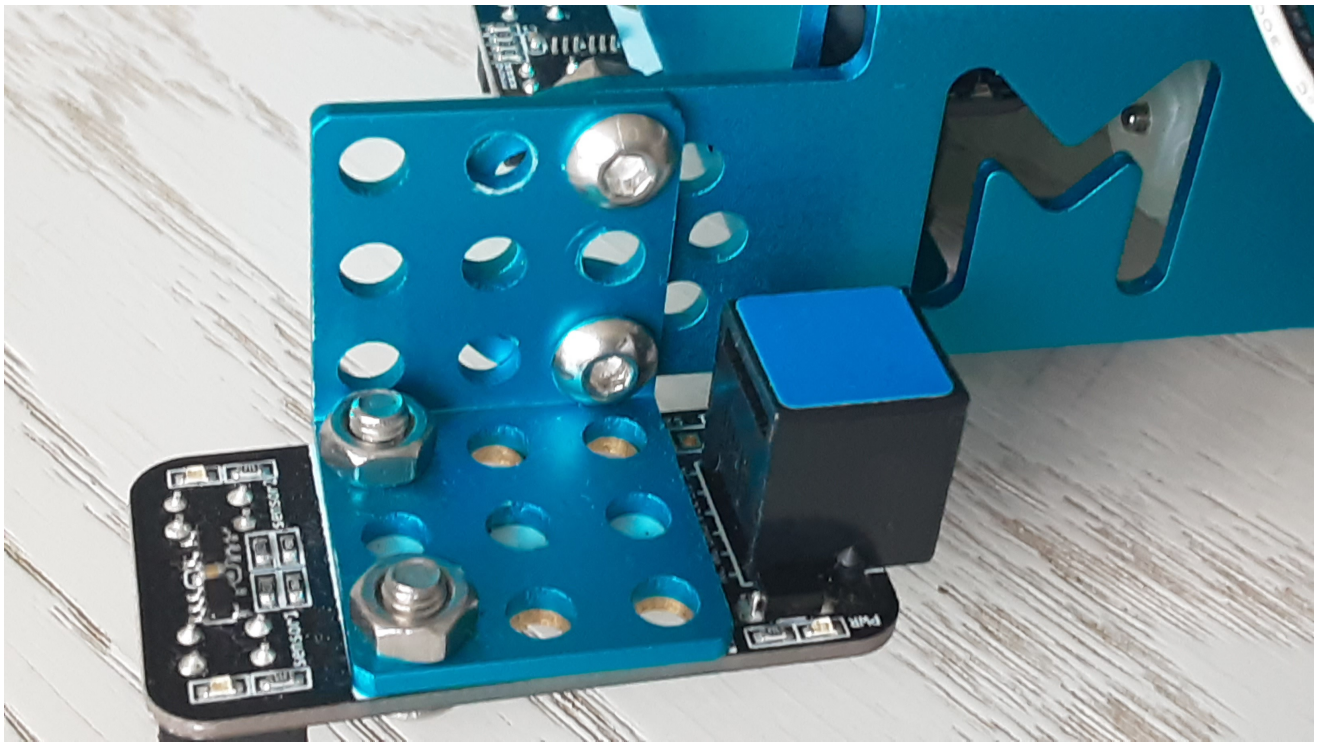
The following version gets by with a line following sensor: Crash prevention with one sensor

Here is the associated programme: Crash Prevention Programmes:

I have extended this programme somewhat, especially by adding a fire brigade sound and the use of the TFT display. That was more elaborate than I thought. The TFT display in particular consumes a lot of computing time and then the timing is no longer right.

The result can be seen here: Youtube video crash prevention

At the first attempt, the crash avoidance did not work. However, the robustness of the mBot became apparent: it survived the crash unscathed. To be on the safe side, however, I then mounted the line following sensors a little differently so that they would detect the edge as early as possible. (In order not to have to screw on the labyrinth, I left a third line following sensor in its usual place. This way, only the corresponding cable needs to be reconnected.



Here is my programme version: Tischkante_TFT.sb2 (75.5 KB)

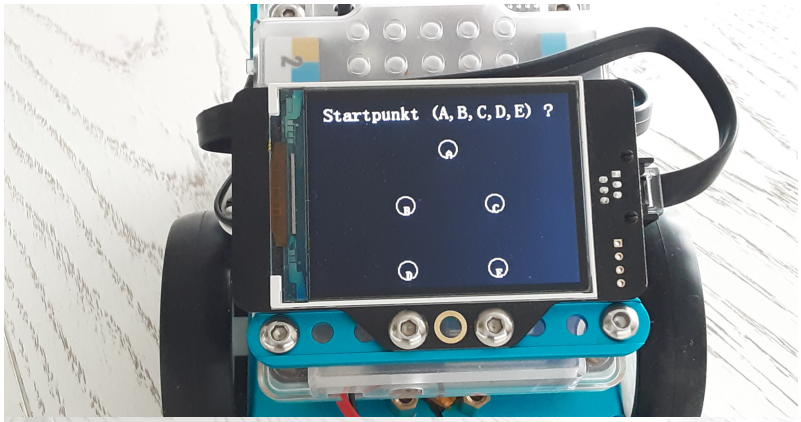
4. the mBot as a computer

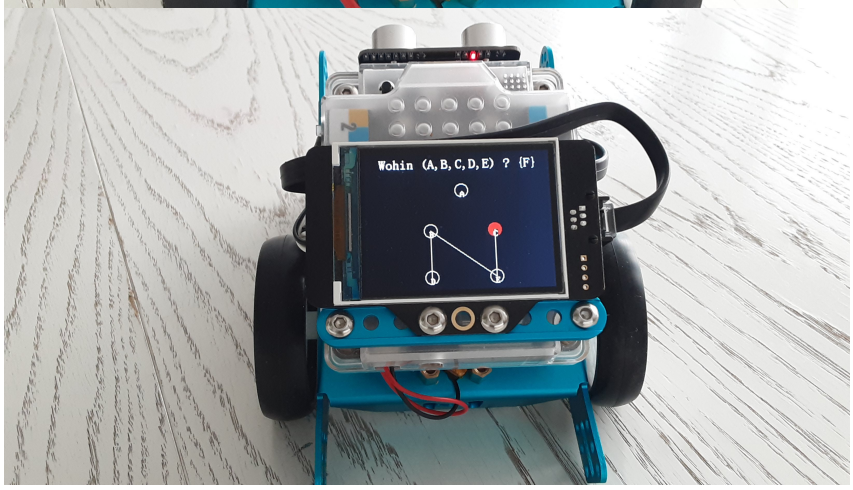
My main interest is to try out how the mBot can also serve as an EDP machine.

I would like to try this out on a series of classic brain games. Almost all of them use a TFT display.

4.1 House of St. Nicholas

Prerequisite: mBot, TFT_Screen, Extensions: AdvancedArduino_Fork, TFT_LCD(arduino Mode Only)





The House of Father Christmas is a widely known drawing game. https://de.wikipedia.org/wiki/Haus_vom_Nikolaus The following graphic is to be drawn in one go. There are many programmes on the internet that determine in particular how many and which possibilities there are to achieve this. http://www.mathematische-basteleien.de/nikolaushaus.htm

A Google search with *Scratch Haus Nikolaus* shows many solutions. Usually it is about the solution being generated on the Scratch stage by the computer. https://physik.osz-biv.de/Informatik/7511/nikolaus.php

A Google search with *mBlock Haus vom Nikolaus* did not come up with any matching results.

When I did a Google search with *Scratch Haus Nikolaus, I got* several hits. But it was always just about the computer drawing the house.

And when I did a Google search with *Arduino Haus Nikolaus,* I didn't find anything either.

Since I have the goal of running all programmes independently on the mBot, I tried my hand at it.

In the first step, I wanted to get by without an additional extension. The whole thing was to be a game. The user should draw the house independently. The computer only makes sure that the rules are followed. It also measures the time needed to solve the problem. The game can be restarted at any time by pressing the F key.

It was clear to me that without arrays it is only possible to check the rules very awkwardly. Painting the picture is still relatively easy with simple variables. Saving the moves that have already been made and especially checking whether a planned move has not already been made is hardly possible.

Therefore, I have resorted to the AdvancedArduino extension (https://www.lab169.ru/mblock/extensions/). Besides many interesting extensions, it also offers two-dimensional arrays. I highly recommend taking a look at the website mentioned. Unfortunately, there is no further documentation besides this, so that for many blocks, despite the many examples, only trial and error remains.

After programming, I made an interesting discovery. Because I found it better to draw on a white background, I used white as the background colour for the TFT display. This also worked very well when the mBot is powered by USB cable. Even immediately after disconnecting from the cable, i.e. in pure battery mode, everything ran wonderfully. The surprise came the next morning: the screen remained black. The moment the mBot was hooked up to the cable again, the expected image was then displayed.

Here you can download both scripts:

Black on white: Father Christmas with white background

White on black: Father Christmas with black background

Obviously, the white background needs so much energy that it can only be displayed when the battery is completely full.

Here is the video: https://youtu.be/YoTYQgbpOxs

A few explanations about the programme:

As is usual in classical commercial computer programming, the EVA principle is realised:

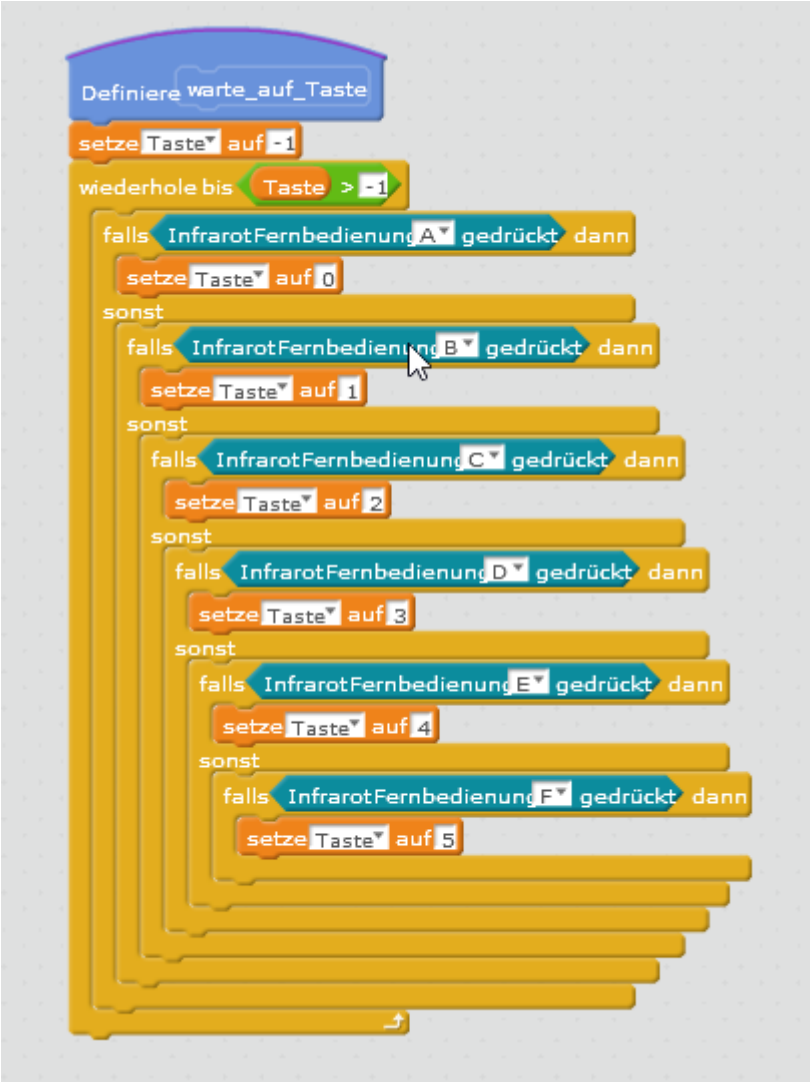
Input

Processing

Issue.

This is reflected in the main programme:

Since processing can only be done when the input is completed, the routine "Wait for key" is used (for details see: ⇒key inputwith the IR remote control).

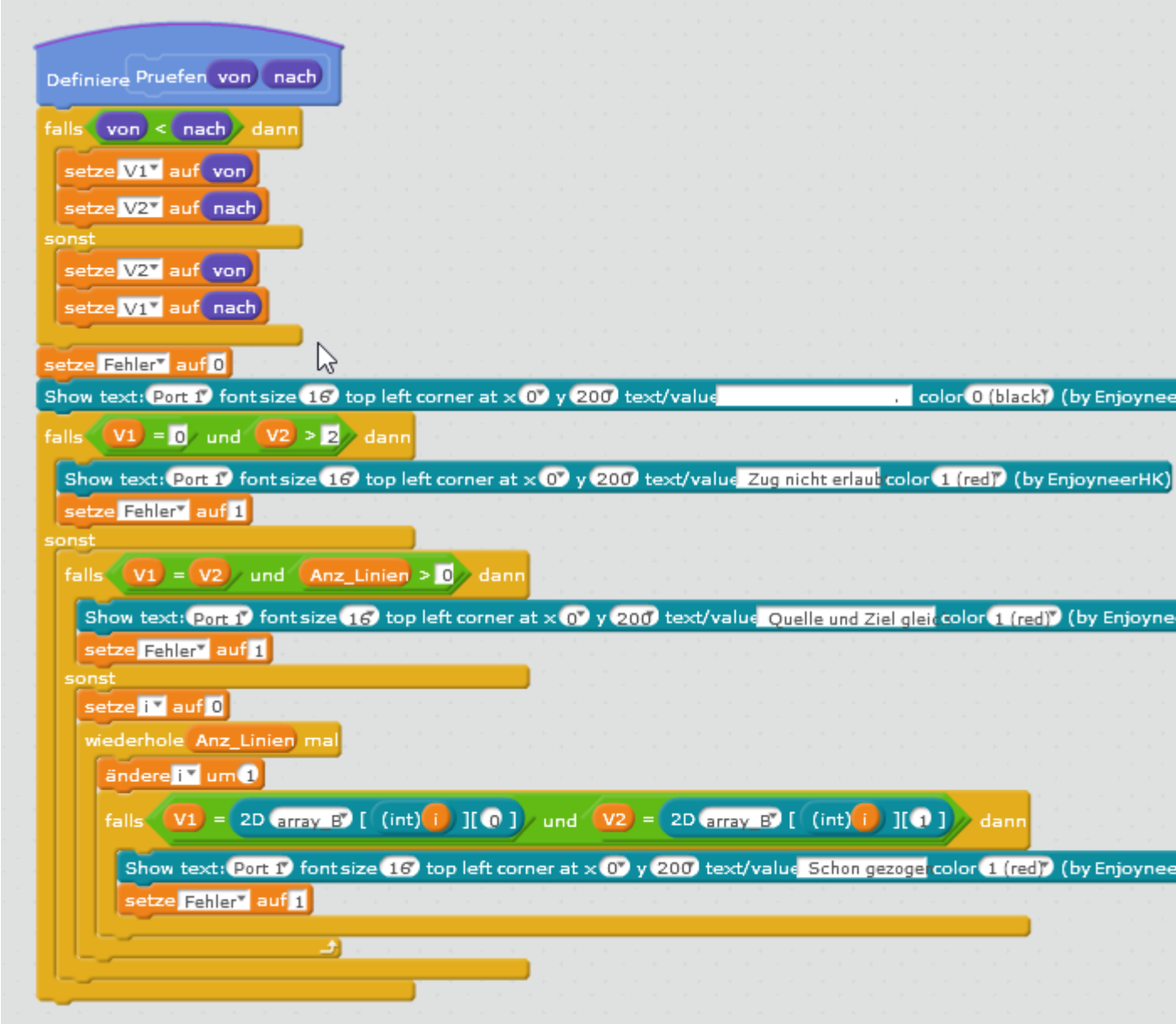


The most difficult part of the programme is checking whether a planned move is allowed.

First of all, do not move to the current location.

Then exclude drawing from the top (point A) to one of the lower corner points (D or E). This is done a little trickily.

Finally, a move that has already been made once may not be made again. For this purpose, all moves that have already been made are entered in a second array. In order to only have to make one check, the smaller of the two points is always entered first. This array is then run through completely. The planned move (again: smaller point first) is compared with the moves already saved.



Finally, it must be checked whether the game is finished, i.e. all permitted moves have been made.

What is still missing is the check whether the player is in a dead end, i.e. no more moves are allowed from the current location. The check routine can certainly be used for this. However, it must be freed from the error outputs.

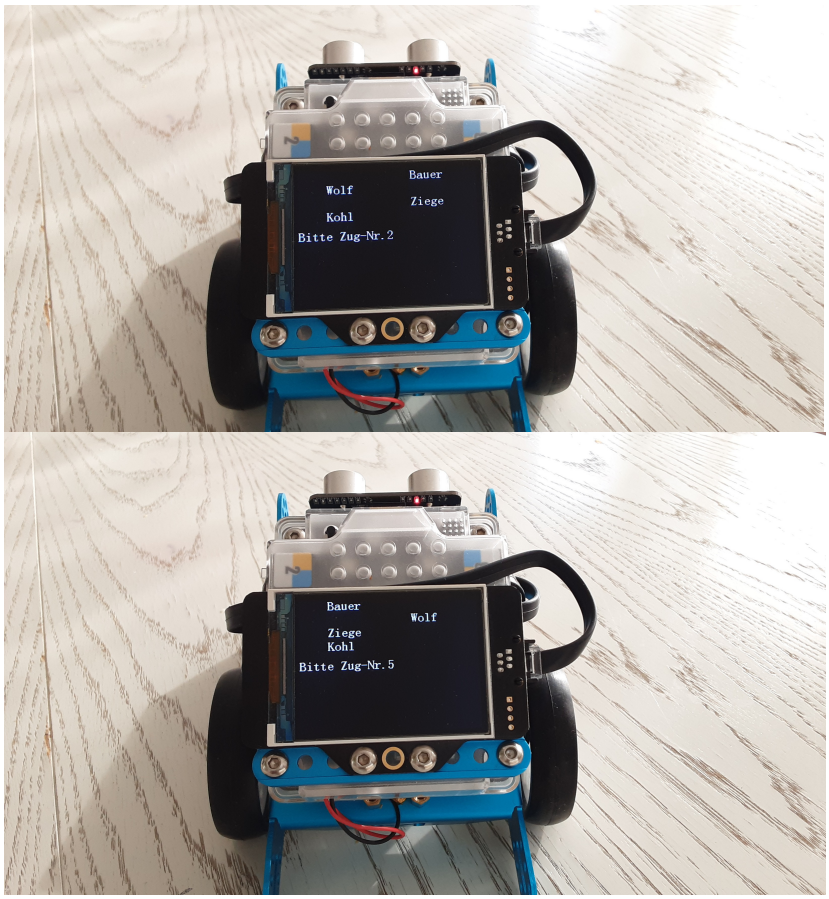
An extension could be that if the end is successful, the five triangles are filled in with colour.

There are a total of 44 ways to draw the house. A programme extension could be that as many different variants as possible have to be found one after the other. To check

whether the ones found differ from each other, a two-dimensional array would probably have to be used.

4.2 Wolf Goat Cabbage Head

Requirements: mBot, LED matrix or TFT screen, graphics for the LED matrix.



This simple puzzle game can be created relatively easily for the mBot.

A farmer wants to cross a river. He has a wolf, a goat and a cabbage with him. The boat is so small that he can only take one object with him. If he leaves the wolf and the goat alone, the wolf eats the goat. If the goat and the cabbage are alone, the goat eats the cabbage.

The input is made via the IR remote control:

By pressing the arrow keys, the farmer drives alone in the boat, by pressing the number keys, the farmer transports the corresponding object.

I have created two variants with the mBot:

The first with the LED display:

Wolf_Goat_Kohl_LED.sb2,

Here is the corresponding video: https://youtu.be/Wpt5uOHpJMI

For the display, the graphics must be located in a specific directory on the local hard disk

The drawings for the LED matrix

For the LED matrix, there is a block that makes it possible to display patterns that you have created yourself.

These are saved by the built-in editor on the local hard disk. Under Windows 10, the path is:

"C:\Users\<username> \Documents\mBlock\emotions\" or in Windows Explorer:

"C:\user\<username> \documents\mblock\emotions\".

For the game Wolf-Goat-Kohl I have created a separate graphic for each of the eight possible positions. These graphics can be downloaded as a ZIP file here: Graphics

By the way, they can have their own name, so the names used by mBlock can also be renamed for better information.

The second variant with the TFT display:

Wolf_Goat_Kale_TFT.sb2

Watch video on YouTube: https://youtu.be/9a7kljDxVe0

Possible extensions: The objects currently "jump" from one bank to the other. It would be nicer if they glided. I could imagine something here for the TFT screen. For the LED matrix, an extension of the extension would be necessary. At the moment, not every single LED can be controlled. That's why the eight different possible combinations of how the objects are distributed on the two banks are realised via eight different images.

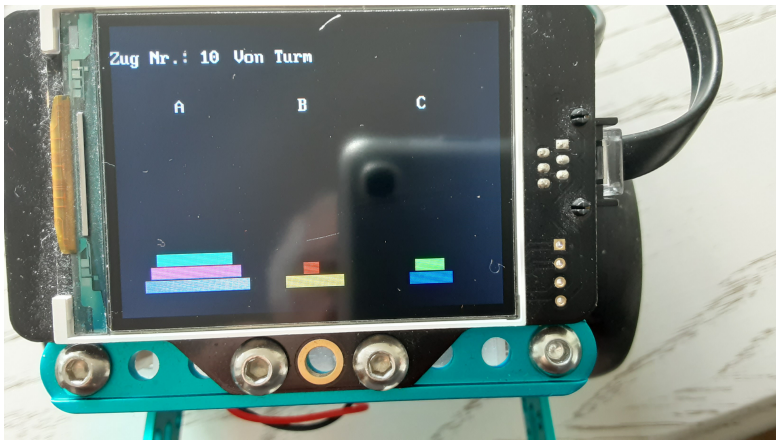
A visualisation of the "eating process" would also be conceivable.

According to WikipediaFlussrätsel, there are several more variants of this puzzle. These could also be translated into a programme.

So that those who don't have an mBot can also try out this game:

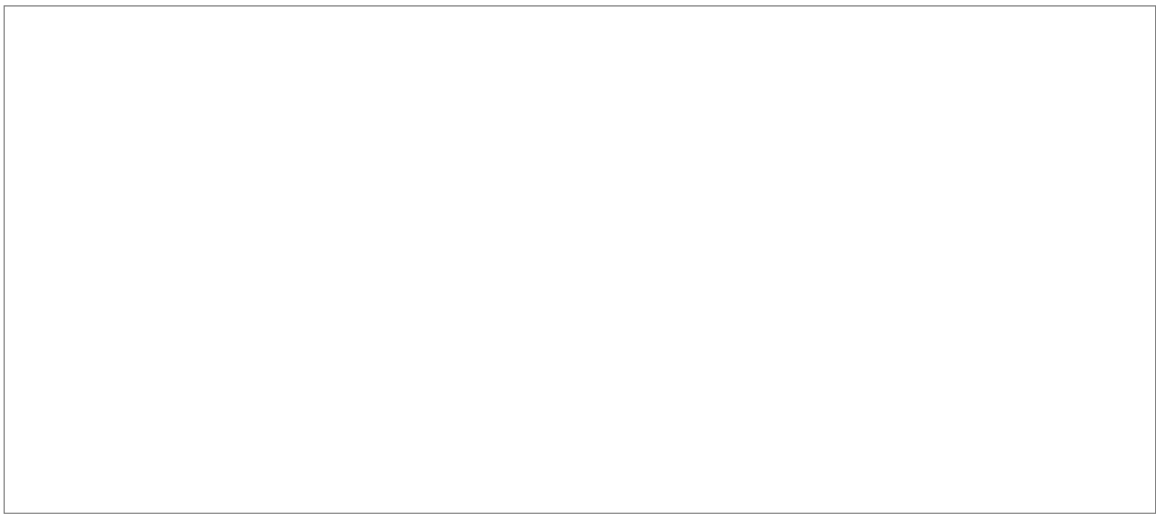
On my server: http://www.prof-horst-guenther.de/spiele/wolf-ziege.

4.3 Towers of Hanoi



The game, which dates back to the last century, is a mathematical puzzle. It is described in more detail on Wikipedia: https://de.wikipedia.org/wiki/T%C3%BCrme_von_Hanoi

It is also known as a game for small children:



Several discs of different sizes are to be stacked on top of each other from one tower to another. Never place a larger disc on top of a smaller one.

In programming terms, there is both an iterative solution and a recursive one. Recursive here means that a procedure calls itself. This leads to very compact programming.

Here is a solution in Pascal:

```
program Hanoi;
type
Tower = (left, middle, right);
const Towers:array[Tower] of string[6]=('left','middle','right'); procedure
MoveStack(const Number: integer; const Source,Target,Help:Tower); begin if Number>0
then begin MoveStack(Number - 1, Source,Help,Target); Writeln('Move slice ',Number ,'
from ',Towers[Source],' to ',Towers[Target]); MoveStack(Number -
1,Help,Target,Source);
end;
end;
begin MoveStack(4,left,middle,right); end
.
```

The programmer's saying "recursive usually goes wrong" is true, but it does not apply here.

The programme exists in just about every programming language.

Rosetta Code (https://rosettacode.org/wiki/Towers_of_Hanoi) lists 162 solutions, plus some sub-variants.

This does not even include one of the many scratch solutions (e.g.: https://de.scratch-wiki.info/wiki/T%C3%BCrme_von_Hanoi) and of course my mBot solution (Hanoi.sb2 (79.4 KB).

Since the Arduino supports recursive programming, I chose this variant.

On YouTube you can watch both the manual and the computer-controlled transport: https://youtu.be/Au7430pcGJI

There are also a lot of offers of this game on the internet. So I don't know yet whether I will create my own version for the internet.

Here are some interesting variations:

Two-click variants:

https://romek.info/games/hf.html

www.gf-webdesign.de/tuerme-von-hanoi/tuerme-von-hanoi.htm

Drag and drop:

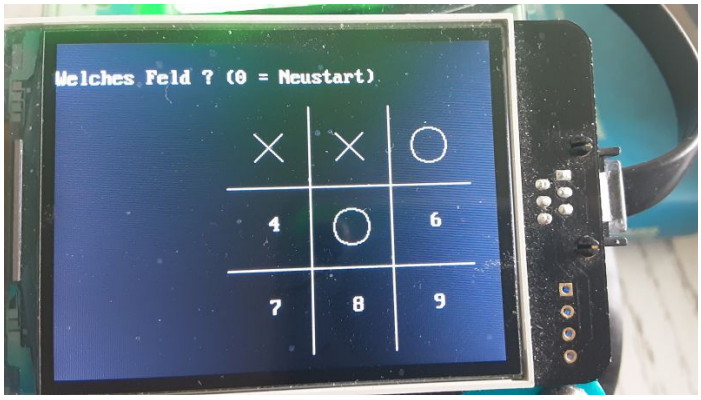
https://www.bernhard-gaul.de/spiele/tower/tower.php

https://de.khanacademy.org/computing/computer-science/algorithms/towers-of-hanoi/pc/challenge-solve-hanoi-recursively

Interesting 1-click variant:

https://www.bv-birlinghoven.de/modules/spiele/tvhanoi/index.html

4.4 Tic-Tac-Toe



Although Tic-Tac-Toe is one of the first computer games, I have never bothered with it. It seemed too simple to me. But that is a mistake. If you Google the *game Tic Tac Toe,* you will get almost half a million hits. Of course, there is also a Wikipedia article: https://de.wikipedia.org/wiki/Tic-Tac-Toe If you search for the *game Tic Tac Toe Strategy, there* are strangely more than one and a half million hits.

Among the hits you will find very different contributions and among them the programme sources for almost every programming language.

The very productive site Rosetta-Code Tic Tac Toe (https://rosettacode.org/wiki/Tic-tac-toe) provides the solution in 62 different programming languages. Scratch and my mBlock solution are not included.

However, I have not found anything for mBlock. And for the Arduino almost exclusively instructions on how to build a game board using micro switches and LEDs.

Mostly, the code for a game in which two human players play against each other is also offered. The programme only makes sure that the rules are followed.

Before I start programming in mBlock for the mBot, I wanted to delve a little deeper. I therefore searched for programme sources in PHP and/or Javascript. The result was not very productive. Some of the solutions were very extensive and confusing.

I liked Dietmar Henke's solution in Javascript best:
https://www.henked.de/spiele/tictactoe/tictactoe.htm

I then contacted Mr Henke by email and asked if I could include his programme here. Since he agreed, his programme follows in the next section.

I will tackle the mBot solution shortly.

The MBlock language is an extension of Scratch. A Google search for *Scratch tic tac toe returns* over a million hits. If you select https://scratch.mit.edu/projects/1425/remixes/,

for example, you get an overview of more than 150 hits. Most of them, however, have the same click graphic; random checks have shown that they are all based on a single source.

For the game against the computer, there is an approach where the computer never loses, no matter who starts. This is the minimax strategy. It is implemented with recursive programming. Simply put, this means that a function (in Scratch: a block) calls itself. For this, it is necessary that local variables are also available in this function. This is not the case in Scratch. In addition, a recursive programme often requires a lot of memory. This is very limited in the Arduino Uno. For these reasons, I have so far refrained from programming the Minimax algorithm for the mBot.

Another reason also speaks against it for me: There is indeed the phrase "He who does not fight has already lost." But against a programme that I know I can't win, I don't even want to try.

In my solution you can set the playing strength: 0 = minimal, 1 = medium.

At playing strength 0, the programme actually works haphazardly. When it is its turn, random numbers between 0 and 8 are generated. The nine fields are numbered accordingly. If the corresponding field is empty, it is occupied, if not, the next random number is generated. The computer is thus quasi blind and does not recognise either obvious winning positions or obvious losing positions.

This is, of course, very unsatisfactory. In stage 1, the strategy is as follows: First, all free squares are checked to see whether their occupancy leads to a complete row for the computer. If this is the case, the square is occupied (attack). The computer has therefore won.

If the programme does not find such a square, the second step is to check for all empty squares whether their occupation by the opponent means that he has won. This square is then occupied by the computer (defence). If this is the only losing square, the computer has not lost for the time being. Especially in the early stages of the game, there is neither a clear winning position nor a clear losing position. Any free field is then blindly occupied (as in level 0) via a random generator.

Another level could use a strategy instead of this random move to occupy promising fields.

For the time being, this is an open extension option.

The script can be downloaded here: tictactoe.sb2

The YouTube video can be viewed here: https://youtu.be/0Z-i2QR6tm8

4.5 Nim games

The Nim game exists in many variants. The internet contains a multitude of descriptions, some of them very "mathematical".

http://wikiludia.mathematik.uni-muenchen.de/wiki/index.php/Nim-Spiel

https://www.alraft.de/altenhein/spiele/nim-spiel/grundlagen.html

In the simple variant (x sticks in a row), the strategy that leads to winning is quite simple and obvious.

For the second variant - several rows of sticks, usually placed in a pyramid shape - the winning strategy is anything but obvious. Mathematically, it is based on a special binary arithmetic. This also makes it relatively easy to programme. The Nim game is said to have been one of the first games to be transferred to a computer.

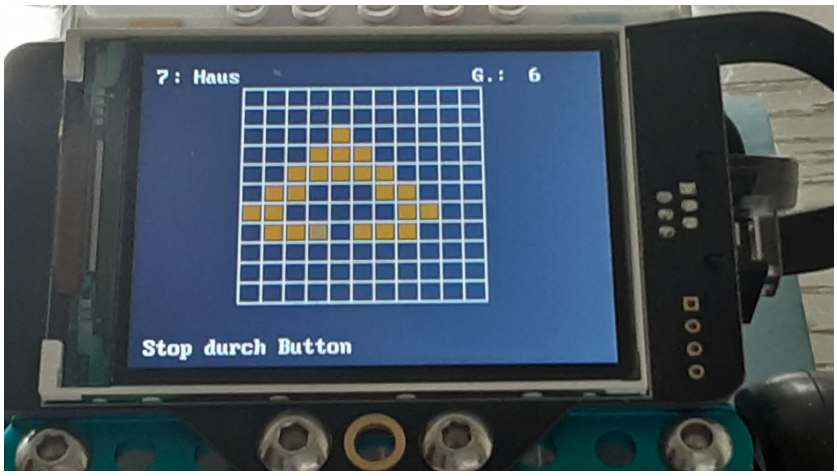
I have implemented both variants.

1. 16 matches in a row, up to three must be taken in turn.
2. A pyramid consisting of four rows with different numbers of sticks. Sticks must be taken from any row; the entire row can also be emptied.

In both variants, the strategy is to get into a "winning position". Once this is achieved, the player can no longer lose if this strategy is followed consistently.

For the computer programme, the problem is to make a decision on how many pegs to take away when he is not in a winning position. Here, the move is then determined by a random function.

4.6 Conway's Game of Life



Conway's Game of Life

In 1970, a British mathematician developed a "cellular automaton" that was hype for a while as the "game of life".

There are "living" and "dead" cells on a rectangular board. Each cell has a total of eight direct neighbours. Any number of generations are created one after the other. If a living cell has fewer than two living neighbours in a generation, it dies of loneliness. If it has two or three neighbours, it survives. If it has more than three neighbours, it dies of overpopulation. A cell with exactly three living neighbours is reborn.

There are a large number of contributions on the internet: A Google search for *Conway game of life* returns over 20 million entries, a search for *Conway game of life* another 310,000.

There are also a variety of computer programs that show this game. Rosetta Code(https://rosettacode.org/wiki/Conway%27s_Game_of_Life) provides more than 170 solutions in various programming languages, including C for Arduino, but of course none for mBlock and none for Scratch. However, there are two projects in Scratch Studio, one of which works well, but not on the mBot.

Here is an instructive example:

https://www.biologie-seite.de/Biologie/Conways_Spiel_des_Lebens

or also:

https://de.wikipedia.org/wiki/Conways_Spiel_des_Lebens

As usual, further useful information and examples are also included there.

I assumed that the implementation would not be particularly difficult. However, this was a blatant misjudgement.

I started with a 16 x 13 pitch.

As far as I can see, there are two different approaches:

The playing field itself is stored in a first matrix. For the sake of simplicity, the matrix has two more rows and two more columns than the playing field. This makes it possible to count the neighbours with a uniform routine for the edges as well.

A second matrix is used for processing. It contains an image of the main matrix. During processing, all surviving cells and the newly born ones are entered into this second matrix. Before the next generation is displayed, the second matrix is copied back into the first and emptied.

In the second procedure, the number of neighbours is entered into the second matrix. This is then used to fill the emptied first matrix again.

There is a small trap: counting the neighbours is sensibly done via two nested loops. However, the central cell for which the evaluation is carried out must not be counted.

During my first attempts, inexplicable errors occurred. When I then displayed the number of gene generations in the upper right corner, I noticed that after the first generation this display either disappeared or strange characters were displayed. When translating with the Arduino IDE, I was told that the memory was low.

Global variables use 1,780 bytes (86%) of the dynamic memory, leaving 268 bytes for local variables. The maximum is 2,048 bytes.
Little memory available, stability problems may occur.

The empirically determined critical limit seems to be 235 bytes. I then systematically saved and also set the first matrix to 12x12.

Another difficulty was that the running time for creating a generation was unbearably long. This was due to the fact that I rebuilt the entire playing field each time: For a dead cell I output a filled rectangle with the background colour, for a living one accordingly a filled rectangle with the colour for living cells.

This output occurred no matter what the cell looked like before. Since most cells are usually dead, this outputted cells in a colour they already had anyway.

I then changed the programme so that before output it checks whether the colour of the cell needs to be changed at all. If this is not necessary, there is no output. Since the output on the TFT screen is very slow, this change resulted in a dramatic increase in speed.

I have entered a total of 10 fixed patterns that can be selected by the numeric keys on the IR remote control when the game starts.

The following interesting patterns can be retrieved:

0: Corner Test1
: Blinker2
: Test3
: r-Pentomino4
: Glider5:
A for All6
: SpaceShip7
: House
8: Kickback9
: Octagon

The fact that the mBot is waiting for an input is also indicated by the green LEDs on the board.

If a run is to be aborted, the button on the board can be pressed until the LEDs briefly light up red and then green.

Actually, I wanted to make the initial pattern editable as well. Since I don't see any way to save such an own pattern, I chose a different approach:

The extension ⇒AdvancedArduino (or AdvancedAdruino_Fork) offers the possibility to

save own function definitions on the local disk of the PC and then integrate them into the programme via an #include statement. Besides the "normal" version(GOL.sb2 (80.1 KB)) I have created another one (GOL_include.sb2 (79.7 KB)). For this one, it must contain a function data that can be created with an editor and stored somewhere on the disk. In the programme, the exact path to the file GOL_Daten_1.inc (6.6 KB) must then be entered in the block def.

4.7 Other candidates for the game

Actually, I also wanted to try to implement *Mastermind* and *Four Wins* on the mBot. However, the experience I had with the low memory space for variables in the Game of Life made me refrain from doing so at the moment.

I don't know yet whether I will implement the following possible candidates. In principle, they would be suitable on the input and output side. However, they are not "mind games".

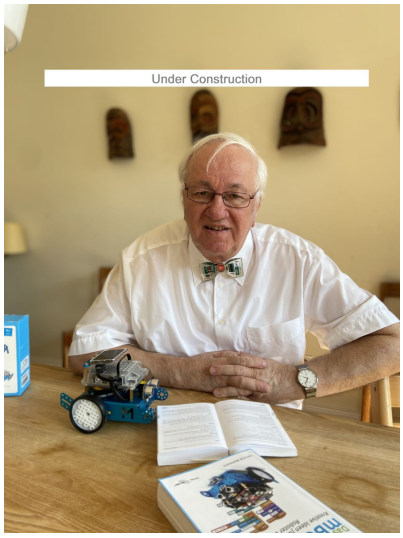
Tetris

Snake

Pong

Again, I fear that there are problems with storage space.

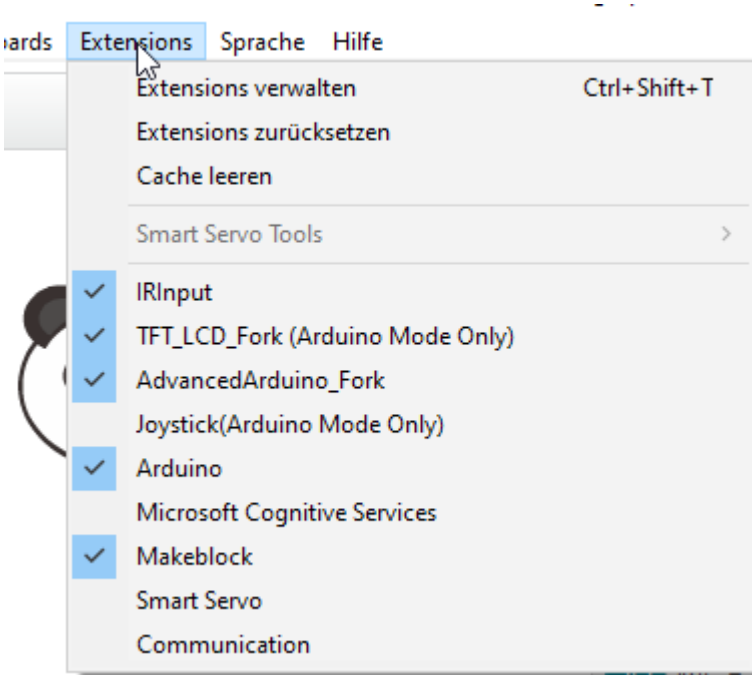
Therefore:



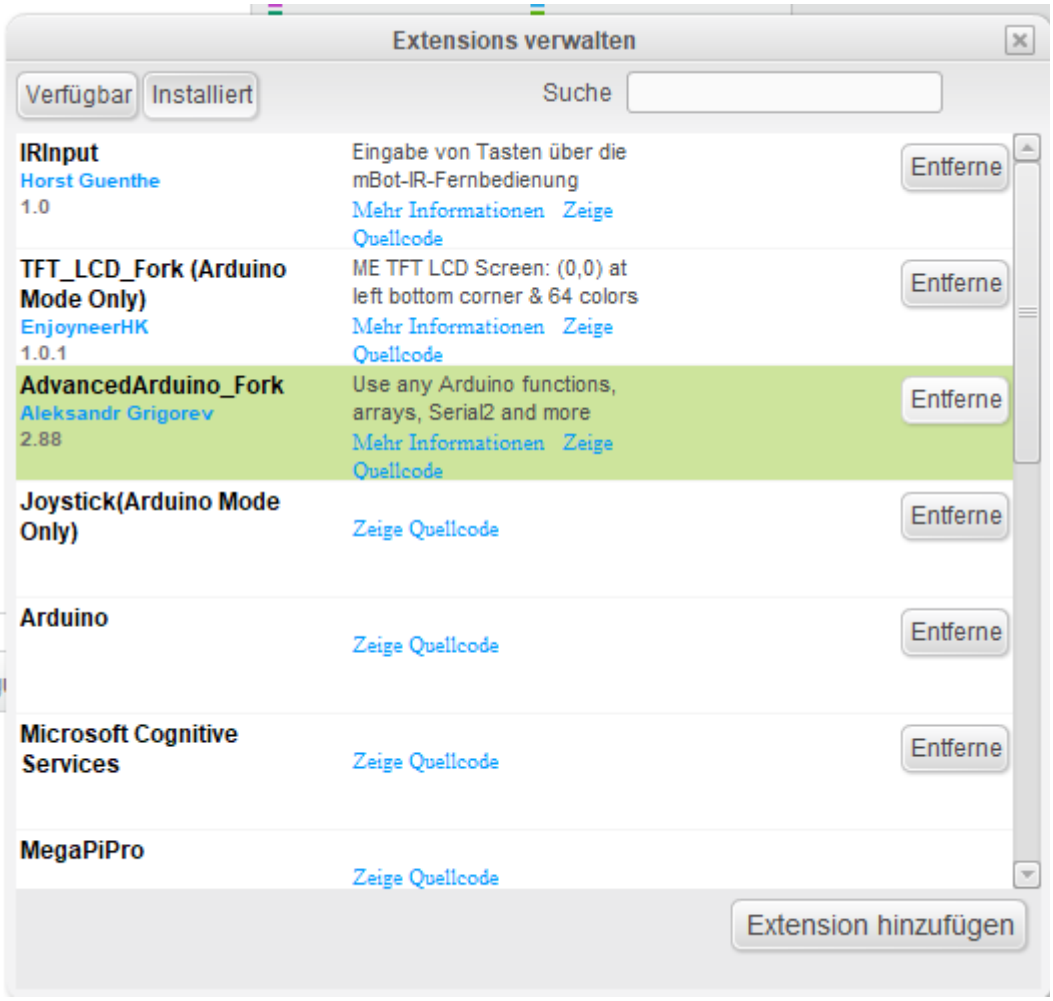
5. extensions

When I first got into MBlock programming, I didn't realise the importance of the extensions. In the meantime, I am enthusiastic.

Clicking on *Extensions* in the mBot menu opens the following submenu:

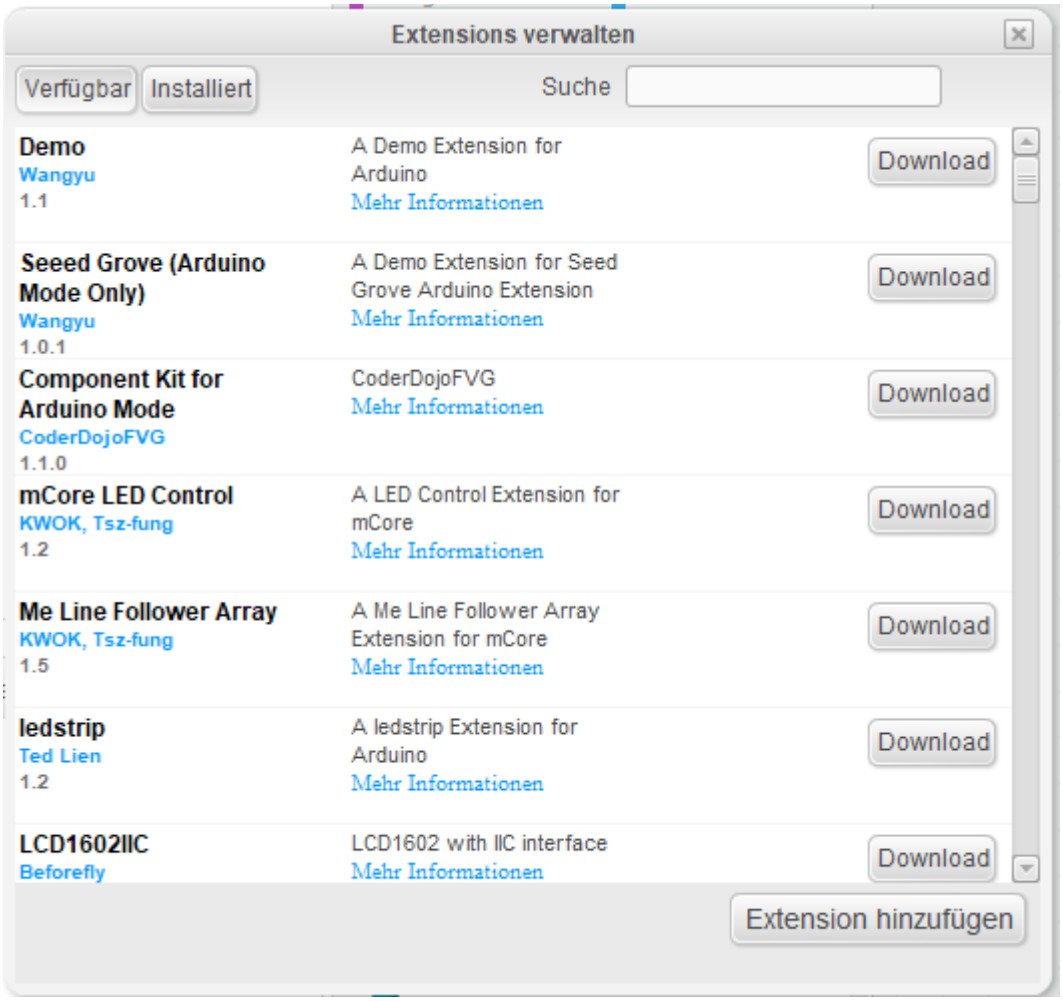


Clicking on *Manage Extensions* opens the following window:



The available and installed extensions are displayed.

Clicking on the *Available* tab opens another window:



All the extensions that are available in Makeblock's Extension Centre are displayed here. Since anyone can upload extensions here, a large proliferation has arisen. I have the impression that Makeblock does not check the extensions as promised.

I have taken the trouble to open each individual extension. Actually, clicking on *More information* should open a web page with more details about the extension. However, this only works in rare cases: Either the page is not accessible (404) or it is in some foreign language - often Chinese. Most of the time, however, there is a link to the page https://www.mblock.cc/docs/create-extensions-for-mblock/. This page explains in general how extensions can be created.

The vast majority of the extensions on offer are therefore not usable.

I have edited two existing extensions and created one new one.

5.1 AdvancedArduino

But there are also some real gems on the web: my absolute hit is AdvancedArduino. This extension, created by a Russian (Alexandr Grigorev), contains just about everything that is missing in the standard MBlock. In particular, up to 26 one-dimensional and two-dimensional arrays can be used. The lists (=one-dimensional arrays) contained in MBlock are only available in scratch mode, not in Arduino mode.

Furthermore, routines are included to convert the type of any variable into almost any other type. This is particularly important when working with arrays: The variables created by MBlock have the type Double. In the output, they are always output with two decimal places. If they are to be used as an index in an array, the Arduino reports an error. The conversion to int provides a remedy here.

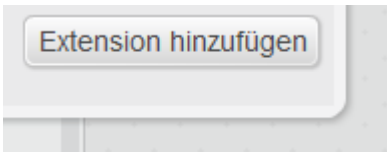
The very special highlight, however, is the possibility to insert your own code. This means that the MBlock language can be extended almost at will. Examples of this can be

found under ⇒Keyboard inputwith the IR remote control**.**

But where there is a lot of light, there is sometimes also shadow: The AdvancedArduino extension is unfortunately only imperfectly documented via many individual examples. Explanations are sometimes completely missing. For example, the last 21 blocks are undocumented and their function is completely unclear to me.

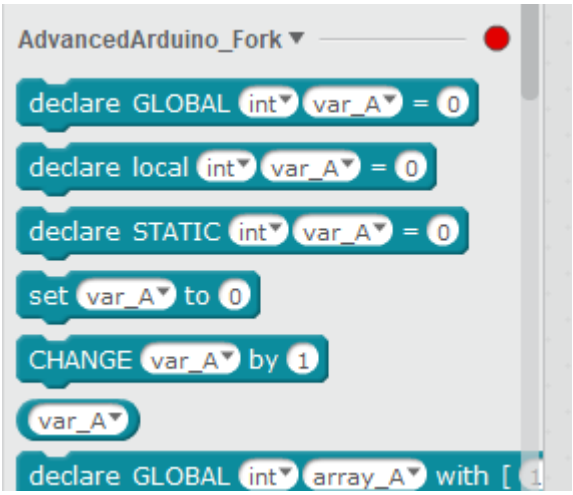
Here is the documentation: https://www.lab169.ru/mblock/extensions/advanced-arduino-extension-c-v/

The very long list makes the use of this extension somewhat confusing. I have therefore created a much shorter extension. This can be downloaded here: AdvancedArduino_Fork.zip (2.9 KB). To install it, go to the menu item *Extensions/Manage Extensions*. When pressing the button



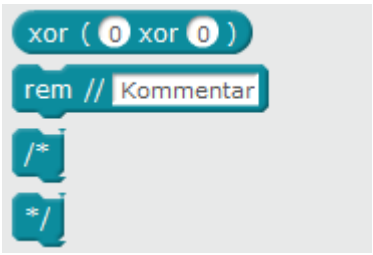
a window appears with which the downloaded zip file can be searched for on the local hard disk. The default setting for the file type must be changed from (*. json) to (*.zip).

Immediately after insertion, the extension is available in the *Robot* section:



For programming the NIM-" (pyramid) ⇒Nim game I needed the bitwise XOR. This is the

^ operator in C++. So that this is also available in MBlock, I extended the extension with a corresponding block (XOR):



I also took this opportunity to add three blocks for comments. There is also a comment function in MBlock, but it is somewhat inflexible in handling. Moreover, it is not transferred to the Arduino sketch.

The // comment serves in particular to structure the source text.

The /* and */ comment can easily be used to "comment out" certain programme instructions, i.e. deactivate them.

5.2 TFT_LCD

Each of the scripts I have presented here in the section *The mBot as a Computer* and some of the robot scripts use the Makeblock TFT display. In order for this to be controlled, a suitable extension must be loaded.

I describe the problems here below in the section ⇒TFT-ScreenSoftware Chaos.

As the most usable solution I decided to use TFT_LCD_Fork (Arduino Mode Only) by EnyoyneerHK.

However, the original version contains a small error in the block *screen direction.* I have corrected this. Independently of this, however, none of the four possible settings seems to have any effect.

A big problem was that the parameter *colour,* which was present in almost all blocks, only worked if a number was given for the colour. Occasionally, however, the possibility is needed for the colour to be stored in a variable. See for example the TFT colour table: Farbtabelle_TFT.sb2 (76.7 KB) To eliminate this error of the creator of the extension (in Hong Kong), almost all blocks had to be edited.

Since I lacked the possibility to create a filled circle, I added this block (*draw a filled circle*).

I have also added another block: If different messages are displayed one after the other at the same place on the screen, there is the problem that shorter messages follow longer ones. If the screen was not cleared beforehand, unattractive text remains are created. To conveniently delete these, I have created the *block Clear line.*

Since this has the same syntax as *Show Text,* it is particularly easy to parameterise.

This corrected and extended version can be downloaded from my website: TFT_LCD_Fork(Arduino ModeOnly)

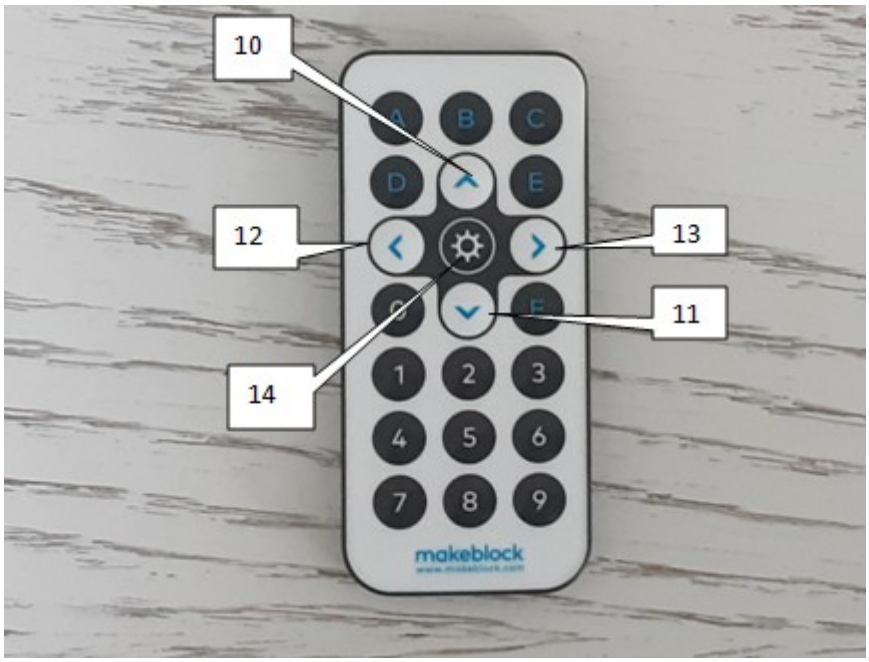
Important note: When switched on, the TFT screen shows the MakeBlock lettering and then the built-in graphics. If the mBot is connected to the PC via the USB cable, there are no problems when the *Clear Screen* instruction is executed: The screen is cleared. However, if the mBot is switched on in battery-only mode, the *Clear Screen instruction*

usually has no effect. This is probably because the serial connection is not yet activated. Therefore, to be on the safe side, the instruction: *Wait one second* should be executed immediately after starting the programme.

You can visit this in the script ⇒House ofFather Christmas.

5.3 IR_Wait_For_Key

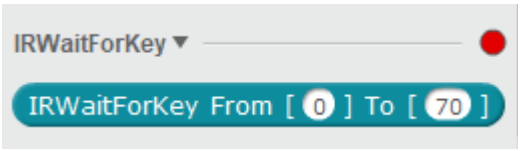
A certain problem with the mBot is the input via the remote control. The extension *IR_Wait_For_Key* simplifies the problem considerably. When the block *IRWaitForKey* is called, the programme stops at this point and waits for one of the 21 keys to be pressed on the remote control. The key code is then output.



The keys A to F provide the ASCII code, i.e. 65 - 70, the number keys their value, i.e. 0 to 9 and the remaining five keys the values shown in the picture.

You can download this extension from my webserver here: IR_Wait_for_Key.zip (1.9 KB)

If it is installed, a corresponding block can be called up in the Robot section:



A simple plausibility check is built in: The valid value range can be declared by entries in From and To:

0 .. 70 = all inputs allowed

0 ... 9 = only the numeric keys

65 .. 70 = only A to F

10 .. 13 = arrow keys only.

The problems and possible alternatives are described in the section ⇒key inputwith the

IR remote control.

There is also a test programme there: Ir_input_with_wait301.3 KB).

6. tools

Sketches that solve certain sub-problems are to appear here bit by bit.

6.1 Keyboard input with the IR remote control

During my first attempts with the mBot, I had big problems with keyboard input. Everything worked wonderfully with the robot scripts, but this was not the case with my first attempts with the brain games. Either the keyboard commands were not recognised or they were executed several times. This was because I had not understood the programming principle of the mBot, i.e. of mBlock.

Classical EDP (from which I come) is characterised by the EVA principle. After a completed input, the processing takes place. And when this is completed, the output takes place. It is therefore characterised by a strictly sequential process.

In technical computer science, e.g. with the mBot, things are different. Here, we don't wait for a completed input, but rather the programme runs merrily along and does some kind of processing or output. In the process, any sensors are continuously queried to see if they have anything to report. If this is not the case, the programme simply continues to work unchanged. If, on the other hand, a sensor has something to report, it switches to its corresponding action. Actually, such a sensor message should trigger a so-called interrupt (i.e. an interruption). However, this is not the case with the mBot: the programme runs in an endless loop and queries the sensors one after the other. However, if it is doing something that is very time-consuming, it does not even query the sensor. As a result, I kept losing keystrokes.

To investigate this more closely, I have created several test scripts. More details on the following pages:

6.1.1. without waiting inline

The first input script works as usual with the many examples for the mBot:

Standard Input_IR.sb2,

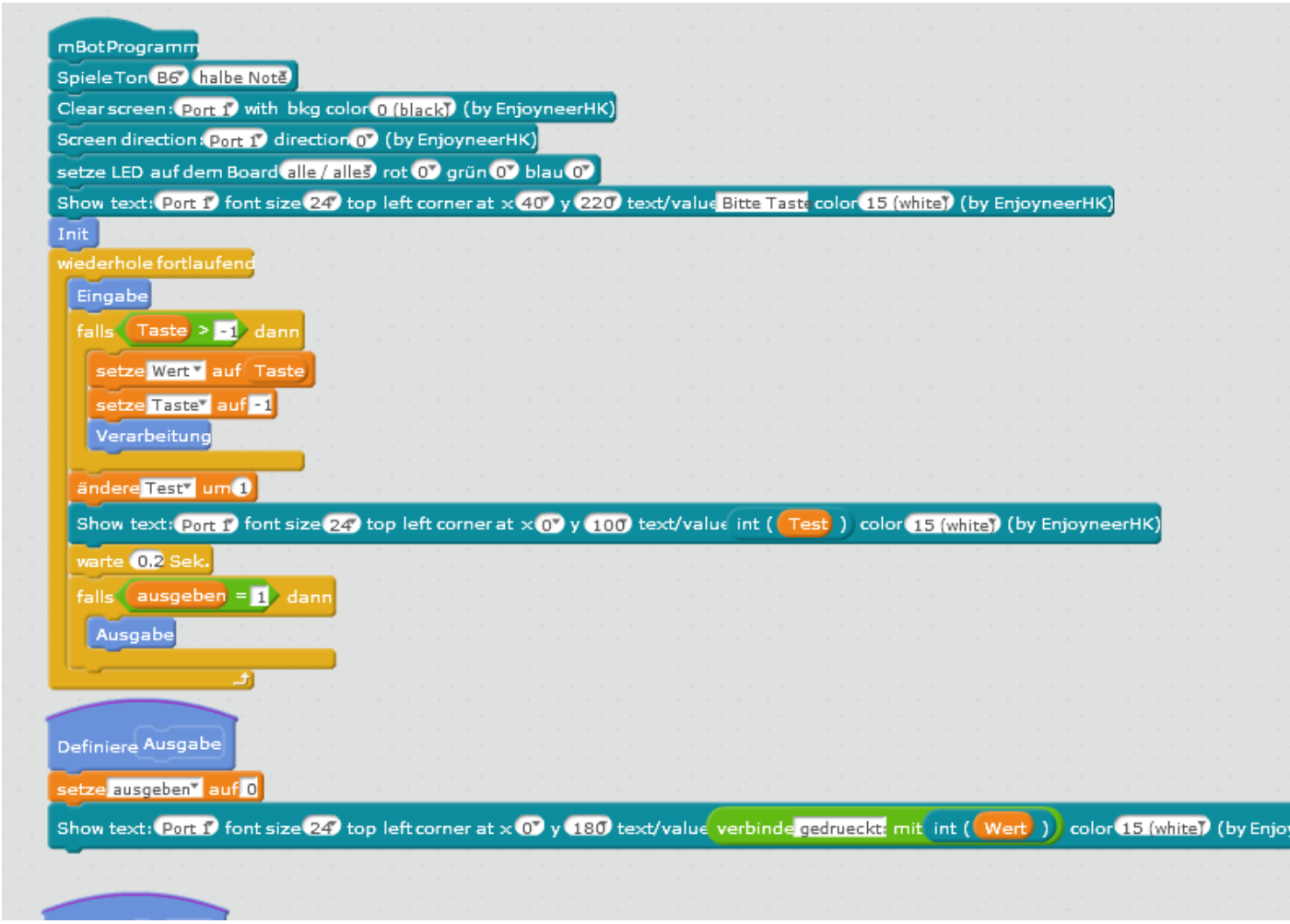
To show that the programme is always running, a variable is incremented and displayed on the screen.

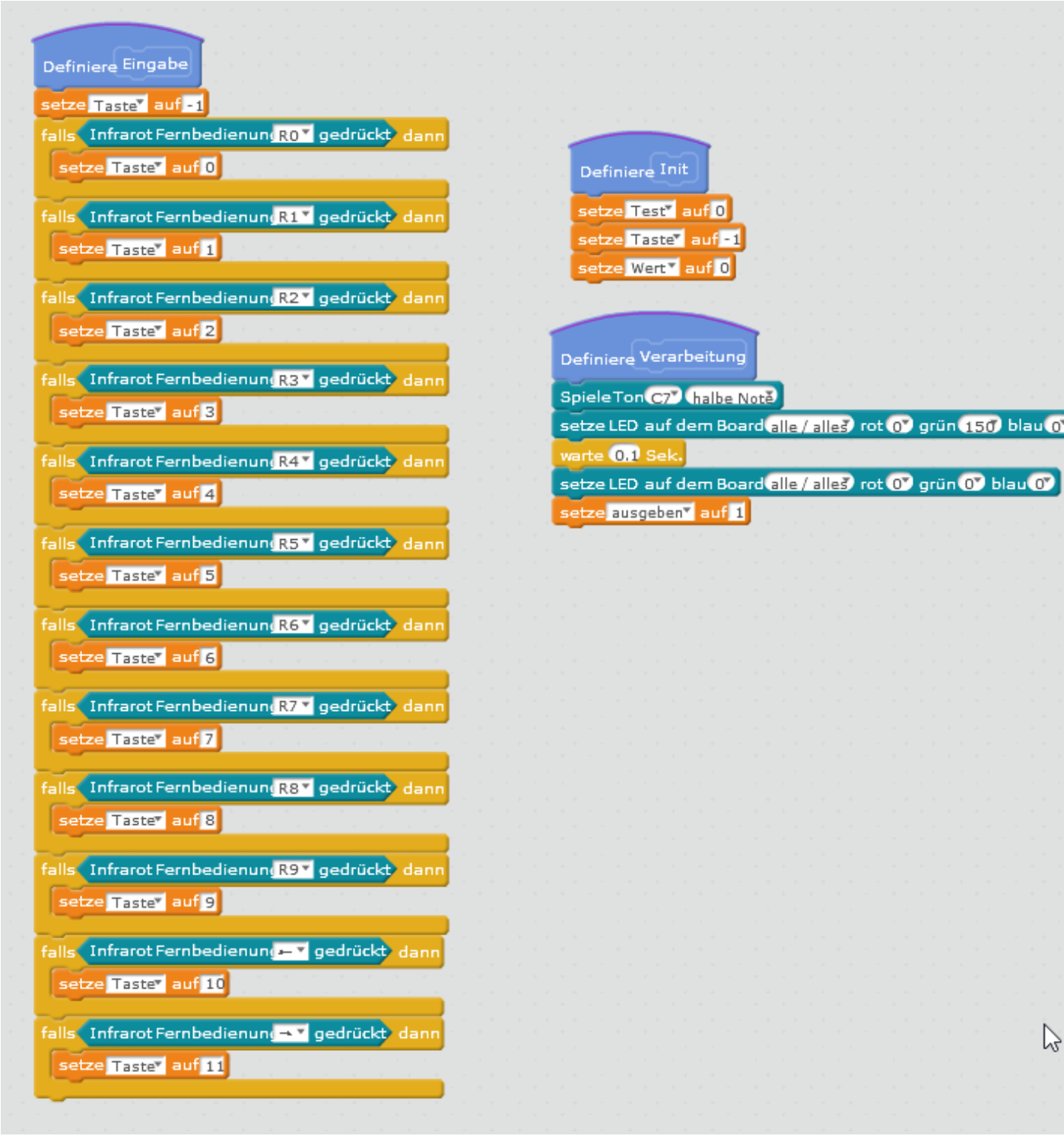
In response to a button press, a short tone is produced and the LED briefly lights up green.

The screen shows which button has been pressed.

However, it happens quite often that a keystroke is acknowledged with sound and LED, but the key is not shown on the screen.

Only the keys 0 to 9 and right and left are evaluated.


mBot Programm
Spiele Ton B6 halbe Note
Clear screen: Port 1 with bkg color 0 (black) (by EnjoyneerHK)
Screen direction: Port 1 direction 0 (by EnjoyneerHK)
setze LED auf dem Board alle / alle rot 0 grün 0 blau 0
Show text: Port 1 font size 24 top left corner at x 40 y 220 text/value Bitte Taste color 15 (white) (by EnjoyneerHK)
Init
wiederhole fortlaufend
Eingabe
falls Taste > -1 dann
setze Wert auf Taste
setze Taste auf -1
Verarbeitung
ändere Test um 1
Show text: Port 1 font size 24 top left corner at x 0 y 100 text/value int (Test) color 15 (white) (by EnjoyneerHK)
warte 0.2 Sek.
falls ausgeben = 1 dann
Ausgabe
Definiere Ausgabe
setze ausgeben auf 0
Show text: Port 1 font size 24 top left corner at x 0 y 180 text/value verbinde gedrueckt: mit int (Wert) color 15 (white) (by Enjo



6.1.2. without waiting (Advanced Arduino)

Among many other interesting possibilities, the AdvancedArduino extension also offers the option of including any C++ code in the script via an include statement.

This code must exist as a simple text file somewhere on the local hard disk.

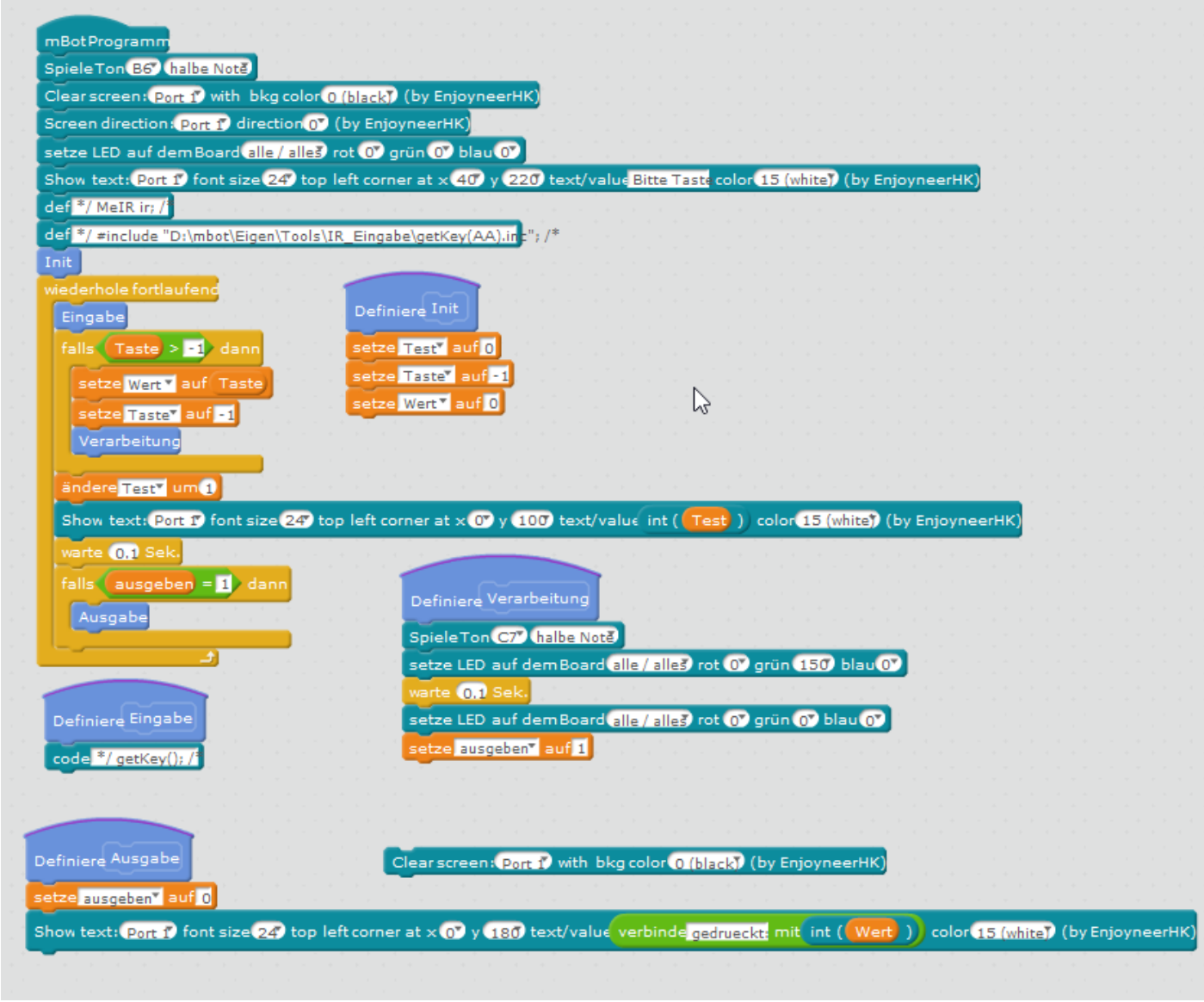
In the following example, the file is called *get_key(AA). inc*

```
void getKey(){ key
= -1; if(ir.keyPressed(22)){key = 0;} if(ir.keyPressed(12)){key = 1;} if(ir.keyPressed(24))
{key = 2;} if(ir.keyPressed(94)){key = 3;} if(ir.keyPressed(8)){key = 4;}
if(ir.keyPressed(28)){key = 5;} if(ir.keyPressed(90)){key = 6;} if(ir.keyPressed(66)){key =
7;} if(ir.keyPressed(82)){key = 8;} if(ir.keyPressed(74)){key = 9;} if(ir.keyPressed(64))
{key = 10;} if(ir.keyPressed(7)){key = 11;} if(ir.keyPressed(21)){key = 12;}
if(ir.keyPressed(9)){key = 13;} if(ir.keyPressed(25)){key = 14;} if(ir.keyPressed(69)){key
= 65;} if(ir.keyPressed(70)){key = 66;} if(ir.keyPressed(71)){key = 67;}
if(ir.keyPressed(68)){key = 68;} if(ir.keyPressed(67)){key = 69;} if(ir.keyPressed(13))
{key = 70;}

}
```

It can be downloaded here: **getKey(AA).inc(0.9 KB)**

The corresponding programme is here: **Standard-Input_IR.sb2 (304.4 KB)**



The main difference is that the input routine is taken over into the programme by the def instruction. The call is then made via a code block from the AdvancedArduino extension.

The advantage is as follows: The input routine required for many programmes only needs to be created once. The problem that under mBlock3 one's own blocks cannot be copied between different projects is thus circumvented.

By the way, the code for the include file can be generated very easily: A normal script is created, which is then transferred to the Arduino IDE. This is then saved as an . ino file and can then be extracted there with a simple text editor and possibly edited.

6.1.3. with waiting inline

In classical commercial data processing, an input routine is usually needed that waits for the input to be made.

Standard_input_IR_with_wait

6.1.4. with Wait (Advanced Arduino)

As already described in the chapter ⇒Withoutwaiting (AdvancedArduino), this input routine can also be integrated via AdvancedArduino.

To create the input module as an external text file in C++.

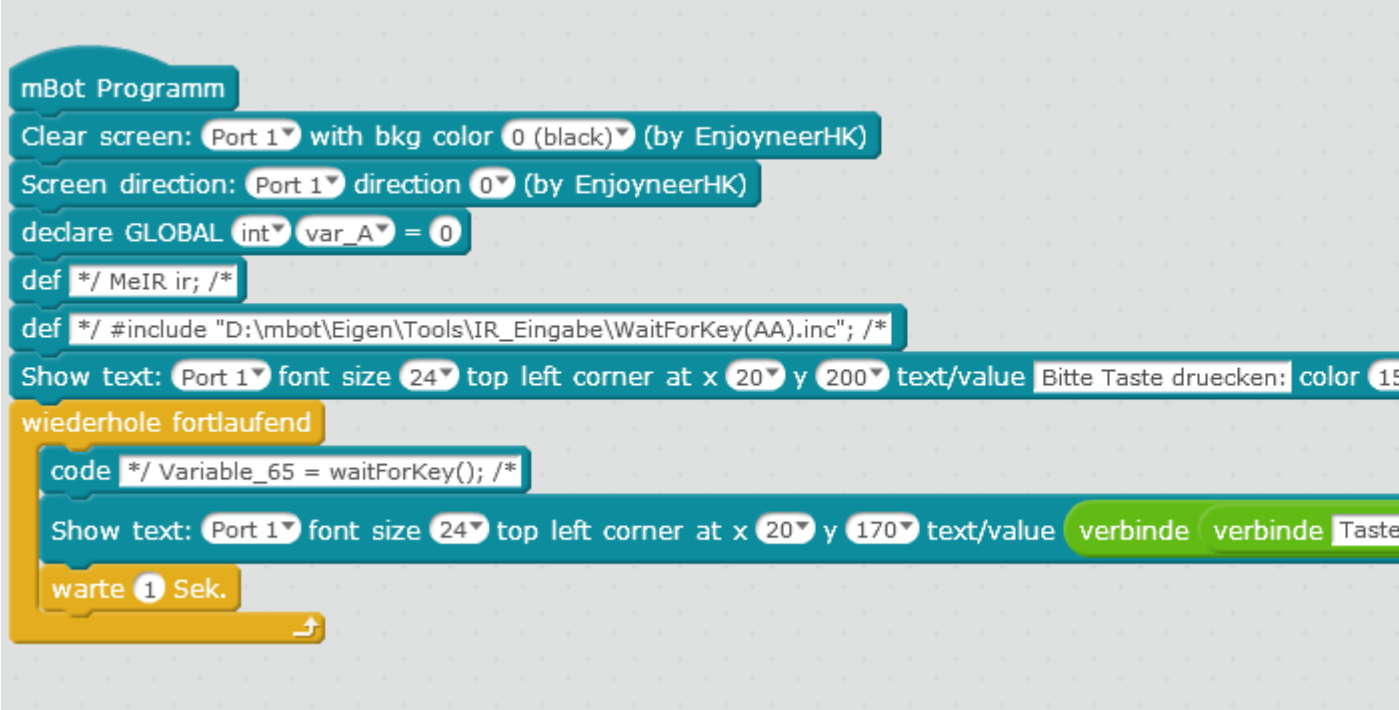
This file can then be stored somewhere on the local hard disk.

It could have the following content, for example:

```
int waitForKey(){
int k;
k = -1; while(!((k) > (-1))){
_loop();
if(ir.keyPressed(22)){k = 0;}
if(ir.keyPressed(12)){k = 1;}
if(ir.keyPressed(24)){k = 2;} if(ir.keyPressed(94)){k = 3;} if(ir.keyPressed(8)){k = 4;}
if(ir.keyPressed(28)){k = 5;} if(ir.keyPressed(90)){k = 6;} if(ir.keyPressed(66)){k = 7;}
if(ir.keyPressed(82)){k = 8;} if(ir.keyPressed(74)){k = 9;} if(ir.keyPressed(64)){k = 10;}
if(ir.keyPressed(7)){k = 11;} if(ir.keyPressed(21)){k = 12;} if(ir.keyPressed(9)){k = 13;}
if(ir.keyPressed(25)){k = 14;} if(ir.keyPressed(69)){k = 65;} if(ir.keyPressed(70)){k =
66;}
      if(ir.keyPressed(71)){k = 67;} if(ir.keyPressed(68)){k = 68;} if(ir.keyPressed(67)){k
= 69;} if(ir.keyPressed(13)){k = 70;}
```

} return(k);
}

The corresponding script would then have to look like this:



So the file is in the following directory: *D:\mbot\Eigen\Tools\IR_Input* and is called *WaitForKey(AA). inc.*

It can be downloaded here: WaitForKey(AA).inc(0.9 KB)

In the script, it is included in a def block of the *AdvancedArduino_Fork* extension with an #include statement.

The function WaitForKey is then called in the infinite loop.

The advantage of this variant over the corresponding own extension is that the effort to create an own extension is omitted and that changes/extensions to the function can be made very easily.

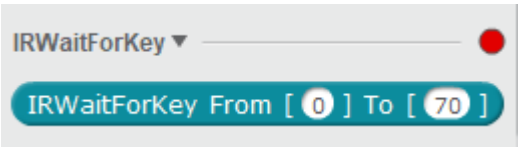
One disadvantage is that this file must exist and be in the right place (mentioned in the script).

The corresponding script is here: IR_input_with_wait(AA (77.2 KB)

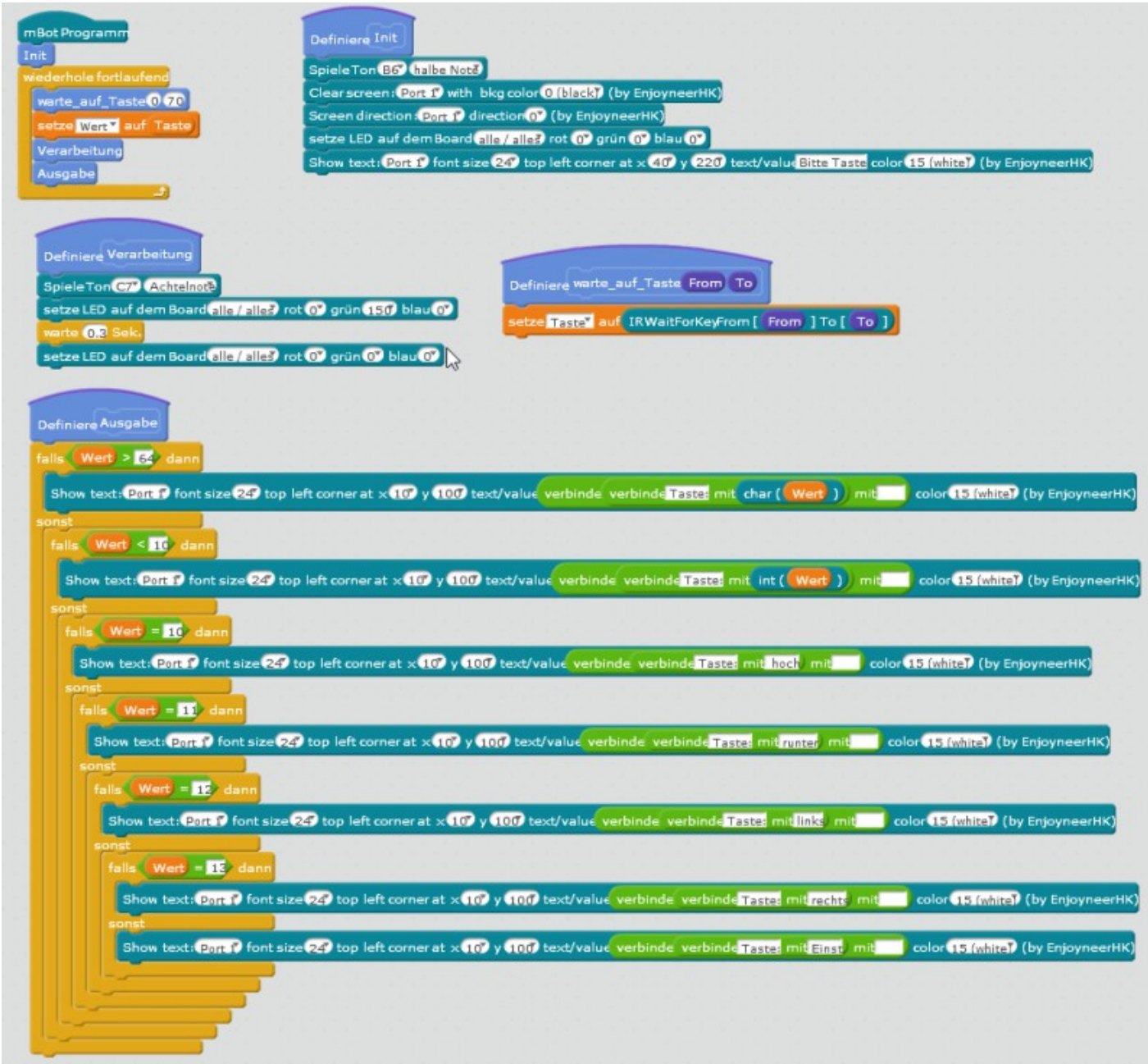
6.1.5. with waiting (extension)

This variant is perhaps the most comfortable.

It uses the extension IRWaitForKey. This must have been installed in the Extensions.



The test script for this looks like this:



The separate block *Wait for key* has only been included for reasons of comparison with the other scripts in this chapter.

As there, each keystroke is acknowledged with a short tone and the lighting up of the green LED.

Here is the corresponding script: Ir_input_with_wait_ext(300.5 KB)

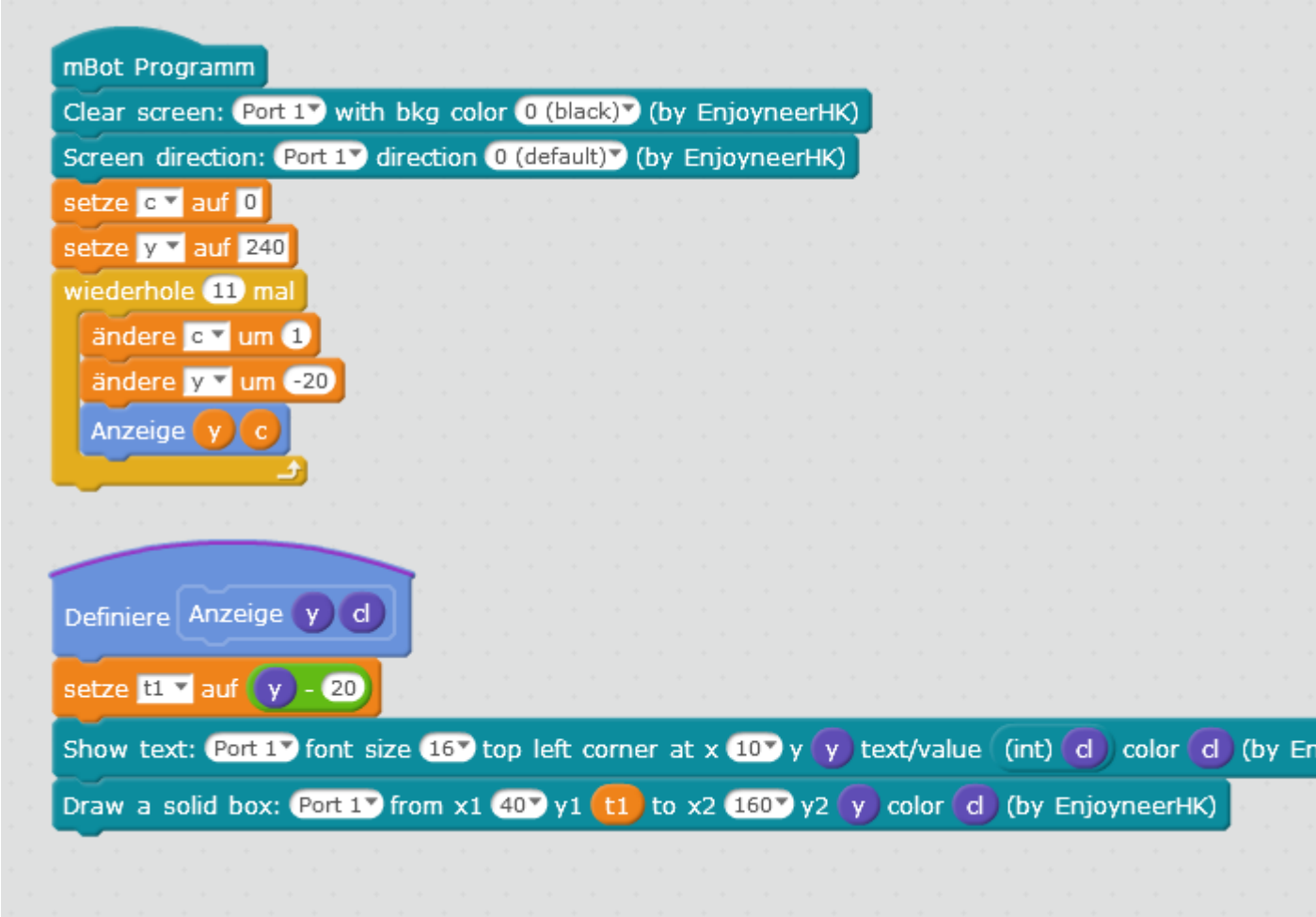
6.2 The TFT colour table

The Makeblock TFT screen can display all outputs in 64 colours. These are identified by a sequential number. However, I have not found a corresponding table on the Internet.

However, the extension TFT_LCD (Arduino Mode Only) by EnjoyneerHK contains a sketch that displays 18 colours in a very small way. However, this is not available as an sb2 file but only as an ino file.

I have created a corresponding scratch sketch. In the process, I stumbled across a very confusing property of the TFT blocks. In all blocks, the various call parameters can also be filled by a variable. Strangely, however, this does not work for the colour parameter. Here, a constant must be selected.

After an intensive search, I discovered that there is an error in the extension. I have corrected it. I have emailed the corrected extension to the developer in Hong Kong, but have not yet received any feedback.



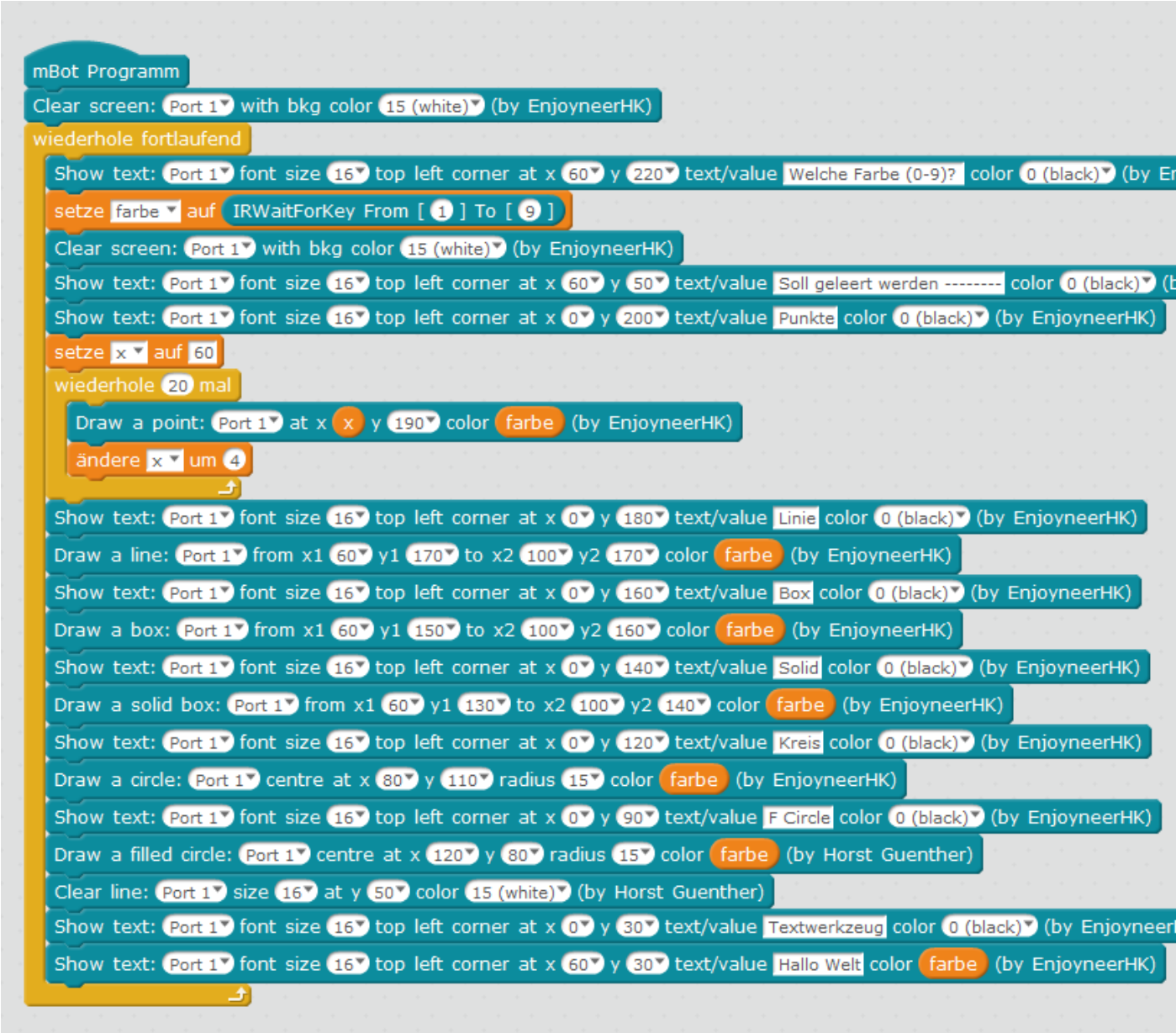
Here is the corresponding sketch: [TFT colour table]

and here the YouTube video: [The TFT colour chart]

6.3 TFT colours and shapes

In order to check whether my error correction has arrived everywhere, I have created a small test script.

In this, all blocks of the TFT-LCD extension are called up:



The associated script can be downloaded here: [Shapes_and_Colours.sb2 (76.7 KB)]

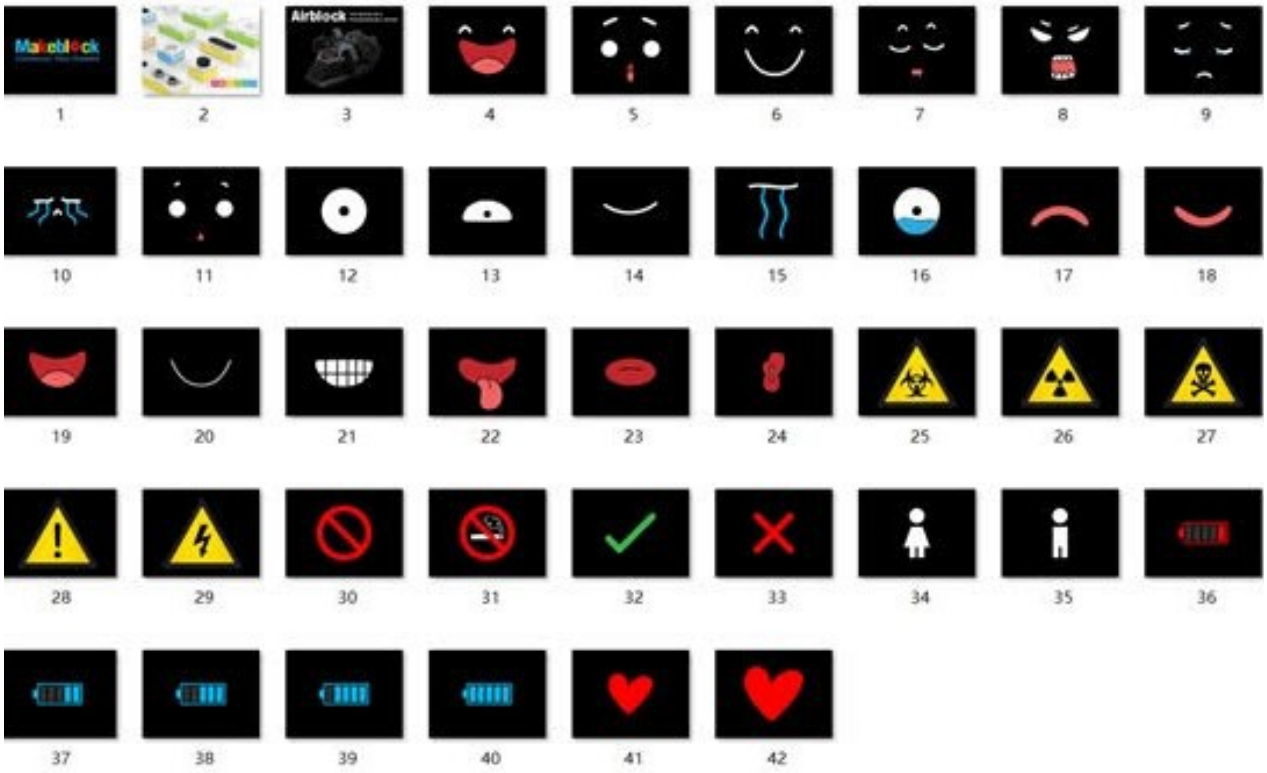
6.4 The TFT images

The Makeblock TFT screen contains 42 pictures. These are displayed in a slide show after switching on. However, there is no possibility to stop this display or even to go backwards.

In the Makeblock forum, a rather small representation of all 42 images is displayed on one page:

https://forum.makeblock.com/t/display-pictures-on-me-tft-lcd-screen-2-4-inch/15860

This graph looks like this:



I wanted to have a way to display these images in a controllable slideshow on the mBot.

For this purpose I have created the following sketch: Picture gallery

After starting the programme, two numeric keys must be pressed. As these are not reliably recognised due to the programme, each successful keystroke is acknowledged with a short tone and a flashing green LED. As soon as two digits have been entered, the corresponding picture is displayed. Caution: A leading zero must be entered for pictures 1 - 9. After a picture has been displayed, it is also possible to scroll back and forward using the keys "<" on the left or ">" on the right.

View on YouTube: TFT picture gallery

What is completely missing is error handling for invalid entries.

The problem with this sketch was that the IR remote control is used as the input device and the challenge was to also be able to enter two-digit values. This is difficult because the Arduino sketch does not run interrupt-controlled and the input is not buffered.

7. problems and wishes

As fantastic as working with the mBot and MBlock is, there are some annoyances. And, of course, there are also wishes for extensions. Since Makeblock has discontinued the further development of MBlock3, it is unfortunately unlikely that they will be fulfilled.

I probably have no choice but to switch to MBlock 5. Of course, this is difficult because almost all the work I have done so far has been wasted.

7.1 Problems

As enthusiastic as I am about the hardware of the mBlock, I am annoyed about various software and documentation deficiencies. The most important aspects of this are described in the following sections. There are also some problems with the extensions, especially the poor documentation. The details on this in the chapter Extensions

7.1.1 Bluetooth dongle

The mBot can be connected to the PC in two ways:

1. About Bluetooth
2. Via USB cable

At the beginning, I thought it would be very practical to connect the mBot to the PC via Bluetooth, especially since the Bluetooth connection to the tablet works very well: When Bluetooth is switched on, you only have to hold the tablet close to the mBot for a short time and the connection is established without any further input. Afterwards, the connection is very stable.

Since my PC does not have a Bluetooth module built in, I plugged an existing ASUS dongle into a USB port. Despite intensive attempts and reading various Internet sources, I did not succeed in establishing a connection. I then got myself another Bluetooth dongle. But that didn't work either.

My enquiry with Makeblock as to whether the original Makeblock dongle was guaranteed to work was unfortunately answered in such a way that they recommend connecting via a USB cable. Of course, I found this very unsatisfactory.

I therefore enquired by e-mail with the company Technik-LPE GmbH (https://technik.lpeshop.de/) whether the dongle they offered was guaranteed to work. As this enquiry was answered positively, I finally bought the dongle from them. And lo and behold: it works perfectly.

In the meantime, however, I have realised that the dongle makes no sense for me:

I don't want to operate the mBot as a terminal device on the PC, but the programme should run directly on the mBot. For this, the programme has to be loaded into the mBot, and that only works via the cable anyway.

Apart from that, the extension I use for the TFT-LCD display only works in Arduino mode anyway.

7.1.2 Incompatibilities between Mblock versions

The operating system world is quite diverse: WINDOWS, Mac, LINUX, IOS and Android. And all of them in many versions. Makeblock seems to claim to support all systems. That is a challenge that is difficult to meet.

And on top of that, there is also a rapid development in Makeblock:

Three different development environments can be considered for the mBot:

1. mBlock 3
2. mBlock 5
3. Blockly

In my own experience, nothing is actually compatible with each other.

The following overview (from Makeblock itself) aims to give an overview of mBlock 3 and mBlock 5:

https://www.mblock.cc/doc/en/mblock3/mblock3-vs-mblock5.html

https://forum.makeblock.com/t/mblock-5-arduino-programming-in-the-former-mblock-3-mode/14308

mBlock 5 is supposed to replace mBlock 3. In any case, mBlock 3 will not be developed further, but will continue to be offered.

Unfortunately, I cannot understand a number of statements regarding mBlock 5: I have not been able to get mBlock 3 scripts to work under mBlock 5. Even the own or third-party extensions available under mBlock 3 cannot be used under mBlock 5. This is probably because they have to be programmed in Javascript for the so-called live mode under mBlock 3 and in C++ for the Arduino mode.

According to the comparison list, C++ is also supported under mBlock 5. However, the Arduino development environment is no longer included and it is also not clear how projects can be saved as . ino files for the Atduino IDE.

Compare the Makeblock Forum post from January 2020:

https://forum.makeblock.com/t/create-a-simple-mblock5-block-for-mbot/15489

Another, but no longer quite up to date, note here:

https://de.scratch-wiki.info/wiki/Vergleich_von_blockbasierten_Programmiersprachen

The projects created under Blockly can be saved and accessed on the tablet via a rudimentary file system. How this happens is not documented. In any case, I have not found these files anywhere in the Android file system. This also seems to rule out the possibility of transferring projects from the tablet to the PC (mBlock 3 or mBlock 5).

In the end, this means that you have to decide relatively early on which basis you want to use: Blockly is very interesting, but it does not offer the possibility of storing the programmes on the mBot and thus making it self-sufficient. I have not found any extensions either. Controlling the TFT screen is therefore probably not possible.

mBlock 3 - which unfortunately is not being further developed - seems to be the most universal system. On the one hand, because the language can be extended almost

arbitrarily via extensions (e.g. the extension ⇒AdvancedArduino), and on the other

hand, because of the direct connection of the Arduino IDE.

mBlock 5 is distinctly cloud-oriented. In particular, the translation of various blocks is missing or not particularly well done.

7.1.3 TFT screen software chaos

Here, too, I have had a lot of unnecessary search work.

The TFT LCD screen is delivered without any written documentation. Even on the web, it is not clear how it can be made to work.

A longer search with Google then yielded the following page:

https://codebender.cc/example/Makeblock/Me_TFT:MeTFT#MeTFT.ino

The solution I tried, however, was unsuccessful. Further searching then led to a post in the Makeblock forum. In it, it was described that only two lines had to be changed:

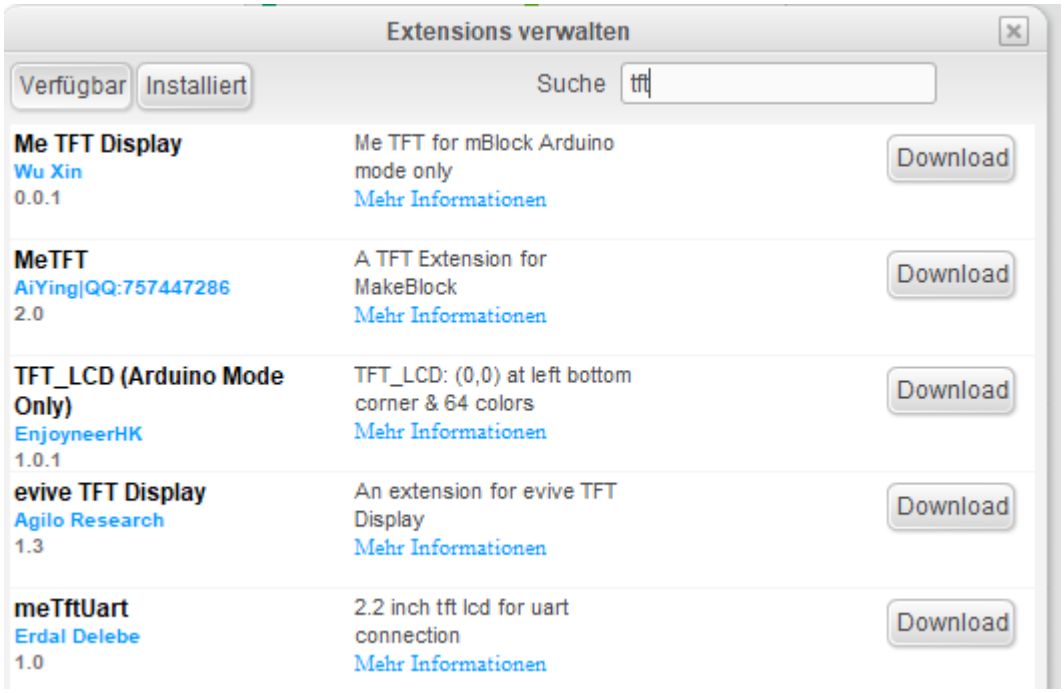
Line 17 of *#define MeOrion_H* to *#include < MeMCore.h>*

Since line 18 is unnecessary but may cause problems, it can be deleted.

In line 21, *MeSerial mySerial(PORT_5);* must become *MeSerial mySerial(PORT_1);*.

With these changes, the script then works in Arduino mode. However, there is no backwards compatibility from this to mBlock. This means that the screen can still not be controlled from there.

Until then, I had not recognised the significance of the extensions. By a lucky inspiration, I then called up the menu item *Extensions* and then *Manage Extensions.*

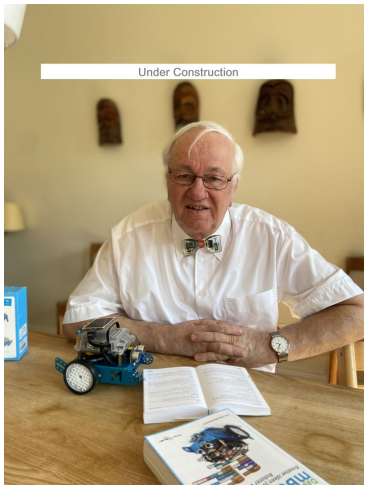


Trial and error has shown that the third (TFT LCD (Arduino Mode Only)) works best.

However, clicking on More information then leads to a page with Chinese characters.

More details on this extension on the page ⇒TFT_LCD

7.2 Wishes



7.2.1 SD card connection

My basic idea is to use the mBot (apart from programming) as a completely self-sufficient device. The annoying thing is that it can only store one programme at a time. There is also the possibility of connecting SD card readers for the various Arduino boards.

I would find it fantastic if one could be connected somehow and then the programmes could be loaded from the inserted card. It wouldn't be so bad if only one programme, a single sb2 file, was stored on each card. This could then be loaded automatically when the card is inserted.

7.2.2 Debugging possibility

From my last development environments (Delphi, ACCESS-VBA, PHP) I am used to searching for errors via a debugger.

Indirectly, the Arduino offers the possibility to make outputs via the serial monitor. However, since the mBot only has one serial port, this is usually occupied, e.g. via the

TFT LED display. In any case, I have not been able to output anything on the serial monitor in parallel. I have not found an explicit description or even an example of this.

With the somewhat confusing project Tic Tac Toe I had a hard time finding actually quite simple errors without a reasonable debugging possibility.

7.2.3 Pairing with smartphone/tablet

The input and output options for my main concern: Especially with the programmes in the section "The mBot as a computer", they are extremely limited. If mobile devices could be used as input/output in some way, that would be very helpful.

7.2.4 Editing sb2 files

At least in mBlock 3, I have not been able to reuse certain blocks from one project in another, when the reusability of modules is an important possibility of modern software development.

Under MBlock3, the sb2 files represent zip archives of JSON files. After unpacking, these can be edited with special editors, but also with simple text editors. However, they have a fixed structure that must not be destroyed.

Purely manually, the own blocks are easily identifiable and can be copied by hand from one file to another. I have succeeded several times, but not always without errors. If a faulty sb2 file is then to be executed, the programme simply hangs without any error message. It is very tedious to find the missing comma/semicolon or the missing bracket. However, it should be easy for the Scratch or MBlock programmers to enable the transfer of blocks.

.

8. videos and downloads

The following table gives an overview of all YouTube videos as well as the download files:

(This is also a good indication of how far I have come in this project so far).

mBot V1

Project	YouTube video	Download sb2 - file
	Robot	
Fire brigade	Fire brigade (LEGO)	Fire brigade.sb2 (76 KB)
Labyrinth automatic	Labyrinth automatic	Labyrinth.sb2
Labyrinth manual	Labyrinth manual	Labyrinth_TFT.sb2
Table edge	Crash prevention	Table edge_TFT.sb2 (75.5
Follow hand	Follow hand	Search_Hand.sb2 (76.1 KB
Reaction test	Reaction test	Reaction test.sb2 (77.2 KB
	Computer	
Wolf Goat Charcoal LED	Wolf - Goat - Cabbage (LED)	Wolf_Goat_Kohl_LED.sb2 (2
Wolf Goat Coal TFT	Wolf-Goat-Kohl (TFT)	Wolf_Goat_Kale_TFT.sb2
Nim game 1	Nim game-1	Nim-1.sb2 (78.4 KB)
Nim game 2	Nim game 2	Nim-2.sb2 (79.7 KB)
Tic-Tac-Toe	https://youtu.be/0Z-i2QR6tm8	tictactoe.sb2
Towers of Hanoi	https://youtu.be/Au7430pcGJI	Hanoi.sb2 (79.4 KB)
Game of Life	https://youtu.be/hv3SxnS4dLA	
House of Father Christmas	House of Father Christmas	Nicholas_BG_s.sb2 (78.1 K Nicholas_BG_w.sb2 (78.1 K
TFT colour table	TFT colour table	Colour table_TFT.sb2 (76.3
TFT colours and shapes		Shapes_and_Colours.sb2 (7
TFT picture gallery	mBot TFT image gallery	Picture gallery_TFT.sb2 (3

Extensions:

Name	Description	Download link
AdvancedArduino_Fork	Universal extension with one and two-dimensional arrays, type conversion functions and the possibility of inserting your own code (original abbreviated)	AdvancedArduino_Fo
TFT-Screen_Fork	Control of the Makeblock 2.4" TFT screen (original corrected and extended)	TFT_LCD_Fork.zip
IRWait	Keyboard input via IR remote control with waiting for key	IRWait.zip (2.4 KB)

mBot V2 (Neo)

Project	YouTube video	Download mblock file
	Robot	
Test Audio Snippet	https://youtu.be/zIIZCD5bdsw	mBot2_Audio_Test.mblock (5
IR remote control	https://youtu.be/RNu612wHb2s	mBot2_IR_Remote_Control.m
Draw House of Father Christmas	https://youtu.be/EuREQtdmZhM	mBot2_draws_Nicolaus_Hou
Labyrinth automatic		
Labyrinth manual		
Crash prevention	https://youtu.be/ZgWDqoD4AuM	mBot2_crash_1.mblock (581
Follow hand		
Reaction test		
	Computer	
Wolf-Goat-Kale IR	https://youtu.be/YrAMQ4dbyo4	mBot2_W_Z_K.mblock (192.4
Wolf-Goat-Kohl Joystick	https://youtu.be/YrAMQ4dbyo4	mBot2_W_Z_K.mblock (192.4
House of Father Christmas	https://youtu.be/pcv1-Zhlrpw	mBot2_Nikolaus_IR_2.mbloc
Nim game 1		
Nim game 2		
Tic-Tac-Toe		
Towers of Hanoi		
Game of Life		

Further downloads:

Name	Description	Download link
Test script for AdvancedArduino	The AdvancedArduino extension offers the possibility to insert any code, even whole functions, into the generated script. Example for this.	WaitForKey(AA).inc(0.
Eight situations for the LED matrix at Wolf-Ziege-Kohlkopf	Graphics required for the game "Wolf-Goat-Cabbage Head in the LED Matrix Version (⇒Wolf-	emotions.zip (2.3 KB)

Goat_Cabbage Head)

9. mBot Games Online

On the following subpages, the games that I have programmed for the mBot can be tried out on the Internet.

Because of the different programming languages and the completely different environment, the programme code is completely different. However, I have tried to make the interface and the operating concept as similar as possible.

In contrast to the mBot programmes, of course, not everything was "written by me". Often I have only programmed the interface or the operating concept myself, the algorithms are usually available on the internet.

9.1 Wolf Goat Cabbage Head

9.2 House of St. Nicholas

9.3 Nim game (one row)

9.4 Nim game (pyramid)

9.5 Tic Tac Toe

9.6 Towers of Hanoi

The Towers of Hanoi

On the page ⇒Towersof Hanoi I have already given links to internet programmes to try

out.

I also like this (English) no-frills version that works with drag and drop:

9.7 Conway's Game of Life

Conway's Game of Life (Game of Life)

This implementation offers (almost) everything the heart desires:

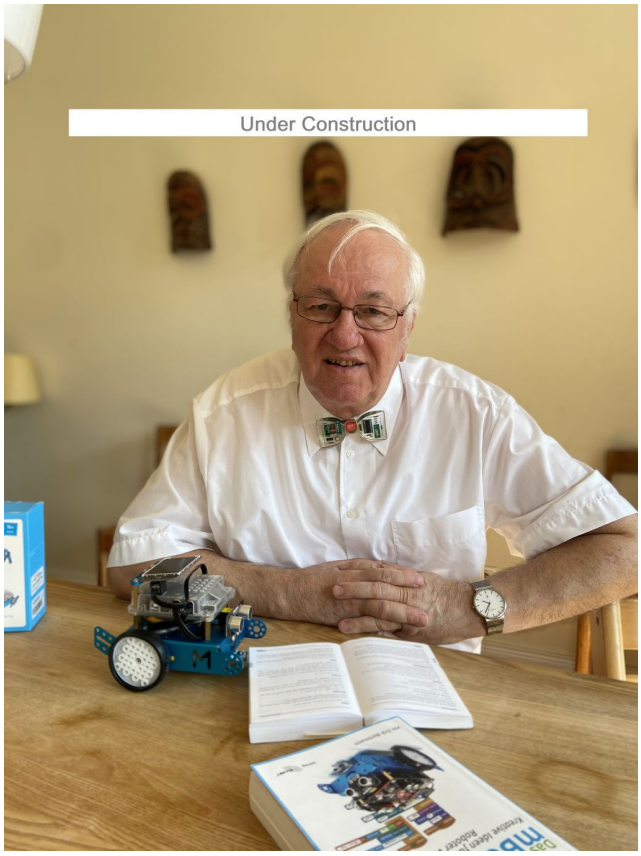
https://bitstorm.org/gameoflife/lexicon

After calling up, the first thing to do is to click on the "Close" symbol at the top right.

10. literature and links

Building instructions for the mBot can be found in abundance on the Internet.

Also a lot of programming instructions.



10.1 Uni Passau: Introductory Script

I really liked a script by Wolfgang Pfeiffer from the University of Passau: Skript Uni Passau There, many introductory examples are explained as tasks with solutions.

10.2 Erik Bartmann: The mBot Book

Bartmann: The mBot Book

Together with the mBot, I got the book by Erik Bartman (Das mBot-Buch, 2016 Germering). It helped me a lot to get started and made the connections clear to me. Bartman explains very precisely the individual steps to build the models he proposes. As an engineer, his suggestions are very technology-heavy, so unfortunately they don't quite correspond to my preferred direction.

His various videos about the mBot on YouTube are highly recommended. Links to these can be found on his website:

https://www.erik-bartmann.de/?Downloads___mBot.

In his book he also gives an internet address for a Facebook group:

https://www.facebook.com/groups/eriksmakeblocktalk/?fref=ts I would have been very happy to join this group, but it no longer exists. Mr Bartmann plans to log off from Facebook and Co. in the medium term.

So far I have found nothing comparable on the internet (apart from the official Makeblock forum

https://forum.makeblock.com/nichts I have found something comparable. The official forum offers solutions to some questions, but unfortunately also to many topics the question but no answer.

10.3 Schertle/Carle: mBot for Makers

Through some internet link I found the book by Schertle/Carle. (mBot for Makers, San Francisco 2017)

I particularly liked one sentence from the introduction: "While the mBot kit and the many accessories available for the mBot are well engineered and made from quality materials, there is a lack of technical support and documentation". (p. IX f.). This corresponds exactly to my experience.

10.4 Lindsay Rooms: MBot and Me

Unfortunately, I only came across this pdf file after a fourth year of working with the mBot:

http://static.education.makeblock.com/mbotandme.pdf

The author has a similar approach to mine: he wants to teach his granddaughter Emma programming with the help of the mBot.

On 273 pages he describes his procedure in great detail. However, he uses MBlock5 and not - like me - mBlock3. However, he has encountered similar problems to mine. Unlike me, however, he has often found solutions to get around these problems.

He also makes very interesting project suggestions. Definitely worth reading.

He also criticises the lack of systematic documentation.

10.5 Miscellaneous links

mBot-Extension Advanced Arduino (Everything your heart desires. Only partially tested so far.

In the Extensions section, I found a very extensive extension from Russia. Here, many possibilities of the Arduino language are also made available via Scratch.

https://www.lab169.ru/mblock/extensions/advanced-arduino-extension-c-v/

11. contact

Since these pages are still under construction, I would be very happy to receive intensive feedback. This could be tips on typographical errors (I am at war with the ß) as well as grammatical errors, but also critical comments and errors in content. Unfortunately, programme errors can never be completely ruled out.

I am also happy to try to answer questions.

Please send messages to the following email address mailto: mbot@prof-horst-guenther.de

12. mBot 2 (Neo)

On the following pages I will describe my experiences with the new mBot 2 (Neo).

In addition to many good experiences, there was also one big disappointment: there is virtually no compatibility with the predecessor, neither in terms of harddware nor

software. Although both the mBot 1 and the mBot 2 can be programmed with a variant of Scratch, there is no way to take over the old programmes.

This is quite understandable, but nevertheless of course very annoying.

In order to familiarise myself, I will gradually create all the programmes I have created for the mBot 1 for the mBot 2 and publish them here.

Differences between mBot 1 and mBot 2

At first glance, both versions look very similar. This is mainly because the chassis has the same dimensions. The new chassis has 12 holes on the side where the "M" was on the old one. The spacing and size of the holes correspond to the LEGO® technology bricks. This makes it very easy to combine the mBot with LEGO®.

The following comparison can be found on https://education.makeblock.com/mbot2/:

Comparison

Specification	mBot 1.1	mBot2
Control board	mCore	CyberPi
Processor	ATmage328 / P	ESP32-WROVER-B
SPI Flash memory	/	8MB
Store multiple programs simultaneously	/	8
Support multi threading	/	yes
Wireless communication	Either Bluetooth or 2.4G (depends on the version) IR	Bluetooth, Wi-Fi and Wi-Fi LAN
Inputs and onboard sensors	Button Reset button Light sensor	5-way joystick Button x2 Reset button Light sensor Microphone Gyroscope-accelerometer
Outputs	Buzzer RGB LED x2	1.44" full color display Speaker RGB LED x5
Expandable electronic modules	RJ25 ports x4, one component per port	mBuild port x1, connect 10+ components i
Additional interfaces	/	2-pin interface x2 3-pin interface x4

In the case of the mBot 2, the improved line-following sensor, the 1.4-inch TFT screen and the built-in battery are not mentioned.

Also left unsaid is that the mBot 1 has an IR sensor with associated remote control, which is incomprehensibly missing from the mBot 2. However, it can (and should!) be purchased for a small amount of money.

There is also an instructive comparison of mBot1 and mBot 2, among others, on the makeblock homepage:

https://store.makeblock.com/products/mbot-neo-robot-python-ai-iot-learning-for-kids

Price-performance ratio

At first glance, the mBot 2, with a price of approx. 140 euros, is significantly more expensive than the mBot 1 at approx. 90 euros:

This comparison alone does not go far enough. To use the mBot 1, at least one battery (20 euros) should be purchased. For my applications : The mBot as a computer, the TFT LCD display (40 euros) is required. This already makes the mBot 2 cheaper than the mBot 1.

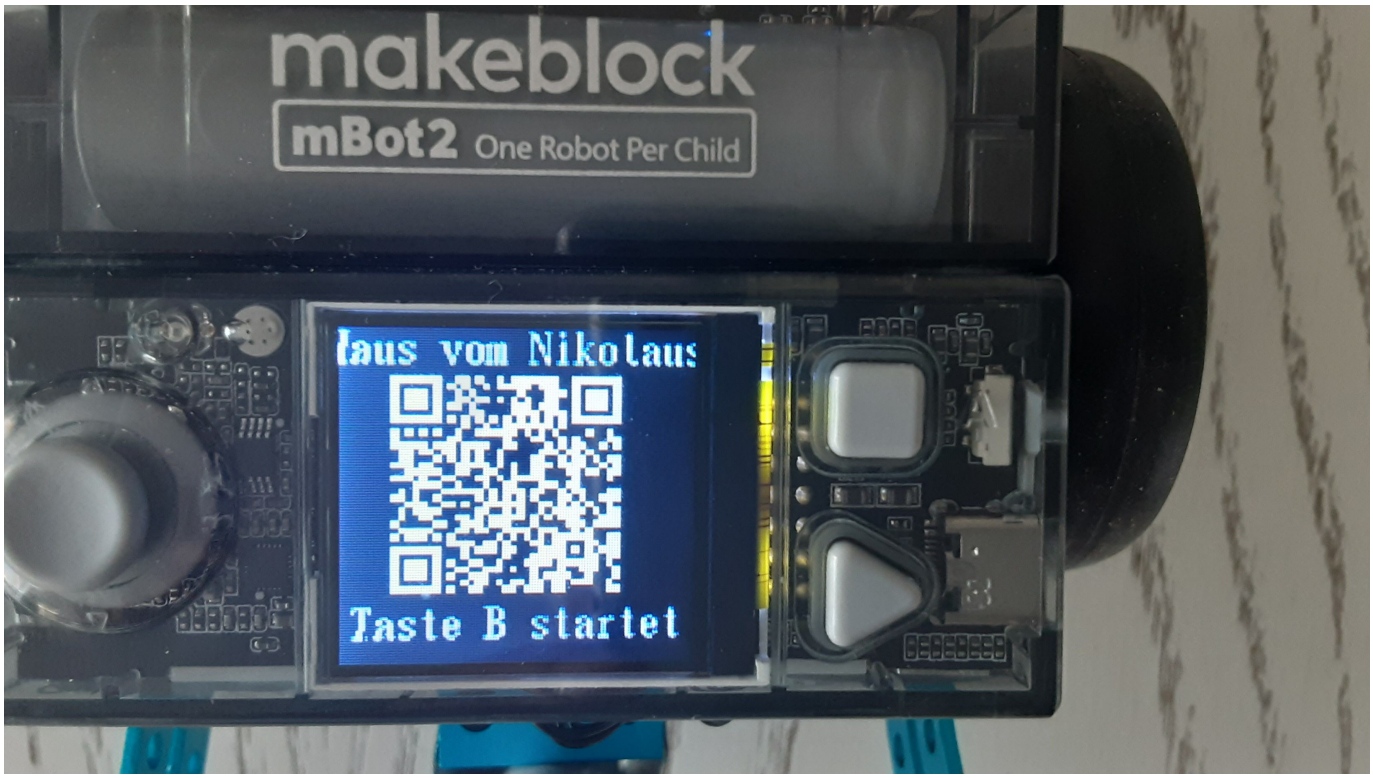
The mBot 2 is clearly superior to the mBot 1 due to the larger number of built-in sensors and the significantly more powerful CPU as well as the significantly expanded main memory and other additions (AI, IoT). The price-performance ratio is significantly better with the mBot 2.

12.1 General experience

12.2. mBot-1 programmes adopted

Since the built-in screen of the mBot 2 is far too small to display the task or the rules of the game, I used an interesting new ability of the mBot:

When all mBot 2 programmes are started, the name of the game and a QR code are displayed. When this is scanned with a mobile device, a special internet page is called up:



A brief instruction is then displayed on this page. It is also possible to branch directly to the detailed explanations. In addition, it is possible to branch to an overview page from which all short instructions can be called up:

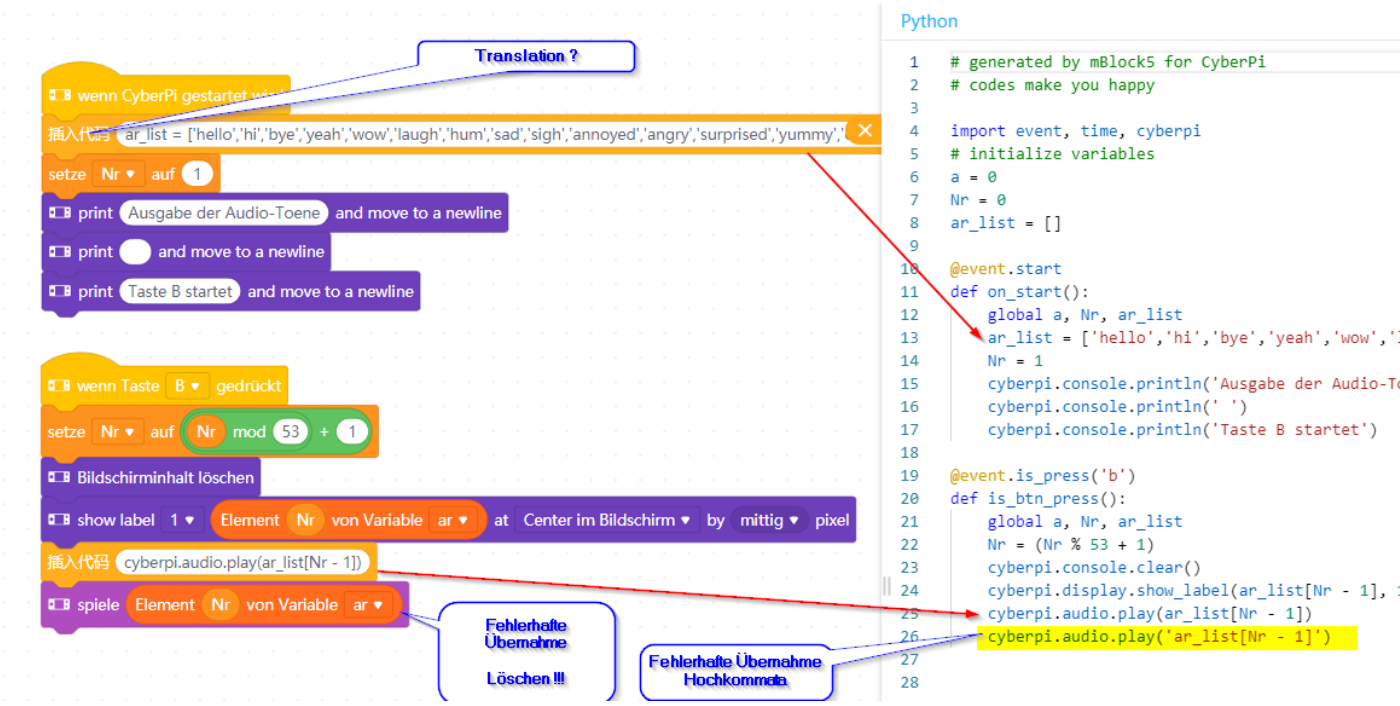
12.2.1 Audio Test Tool

An interesting programme in several respects is the following:

With the mBot 2, there are two blocks in the "Audio" section with which 53 different sounds can be played. However, I have not found an easy way to listen to them. A small live programme must be written that plays a sound. Then the next sound must be selected laboriously via the rather long drop-down list and the sound is then called up by clicking on the green flag.

I wanted this to be more comfortable.

I have therefore written a small programme in which all 53 tones can be called up one after the other by pressing the B key. The (English) designation is displayed in each case and the tone is output.



It is also interesting to use the (untranslated) block to transfer any Python commands into the generated Python programme.

Note: At the end of June 2021 (version 5.3.0), there is still a programme error in MBlock. The "play" command puts the variable in inverted commas, thus turning the variable into a constant. To get around this, I simply used the untranslated command.

Download the programme:mBot2_Audio_Test.mblock (55 KB)

Youtube video: https://youtu.be/zIIZCD5bdsw

12.2.2 Remote control

The infrared remote control, which was still included with the mBot 1, is unfortunately no longer included with the mBot 2.

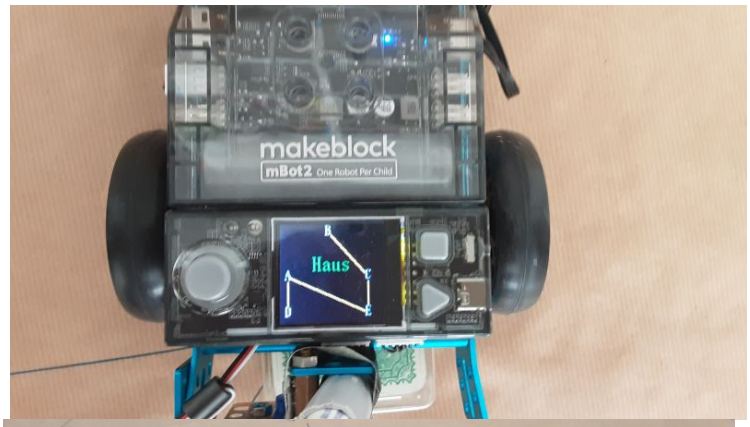
However, an IR transceiver can be purchased separately. The remote control of the mBot V1 works with this, but must also be purchased if necessary.

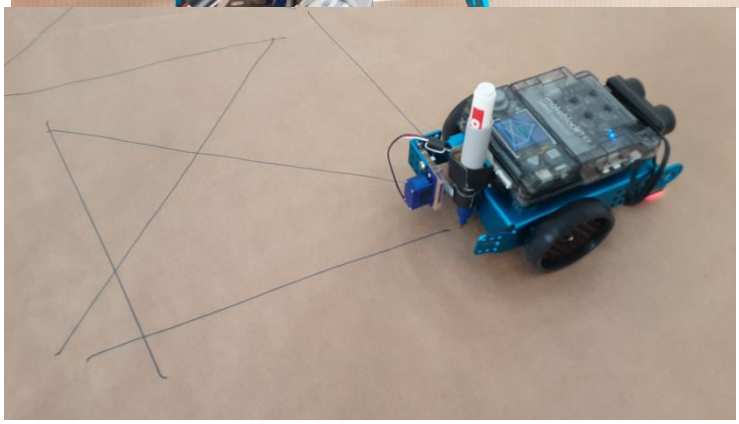
The programme largely corresponds to the "Fire Brigade" programme of the mBot 1

Download : mBot2_IR_RemoteControl.mb (239.4 KB)

YouTube video: https://youtu.be/RNu612wHb2s

12.2.3 Drawing Machine





Inspired by the programme "House of Father Christmas", I came up with the idea that the robot could also manage this task.

A square with two diagonals and an attached right-angled triangle must be driven.

So there are only three different angles (45°, 90° and 135°) and three different lengths to consider. If the side length of the square is fixed (20 cm), the length of the diagonals (28.28 cm) and the roof (14.14 cm) can easily be calculated using the Pythagorean theorem.

This was relatively easy to programme. Makeblock advertises that the new motors of the mBot 2 are much more precise to control. That is true, but not 100%.

The robot follows the programmed routes cleanly, but it is difficult to tell whether it is doing so accurately.

Therefore, I have extended the programme to show on the screen which route he is currently travelling and, in addition, the syllables of the corresponding saying (This is the house of Ni-ko-laus) are also displayed.

But I was not yet completely satisfied with the result.

I therefore came up with the idea of recording the distance travelled on the pad. To do this, I fitted a thin Edding 400 pen with a Tesaband sleeve and attached a guide made from a shortened insulin pen to the mBot. Raising and lowering the pen works very well. This is necessary so that the turns are not drawn.

However, the result was initially very unsatisfactory: there were gaps of about 6 cm at the corners. These gaps are caused by the fact that the pin is placed a good 5 cm behind the mBot's pivot point. The solution was quite simple: The mBot moves 5 cm further with the pin lowered, then the pin is raised and the mBot moves back these 5 cm. Then the pin is lowered again.

The result is not perfect but adequate.

By the way: To make it easy to insert and remove the pin, the servo motor can be moved to the appropriate position via the joystick. (See video)

In the meantime, I am thinking about possible extensions: So far, the mBot drives completely autonomously. However, it would be easy to expand the remote control programme (remote control) so that the pen can be raised and lowered via two additional buttons on the IR remote control. This would then create a somewhat idiosyncratic plotter. With some effort, this could create any drawings.

And that leads me to another extension possibility: back to the House of St. Nicholas!

The control programme could also be designed in such a way that a separate block is created for each of the three turns and the three travel lengths, plus one block each for raising and lowering the pen. These 8 commands could be assigned to certain buttons on the remote control. Pressing the button then triggers the corresponding movement. For the sake of simplicity, the forward and backward movement by 5 cm is carried out at the same time for the straight sections.

This way, Santa's house could also be drawn manually via the remote control.

And this could be topped:

1. The individual keystrokes could be stored in a list. After completing the sequence, it could be retrieved by pressing a button or using the joystick. The mBot has then learned something.
2. It would also be conceivable to store one or more such lists in the mBot, perhaps even to make them readable from files via an extension. It would also be conceivable that stored keystrokes could be output to a corresponding file and read in again later.
3. The high school would be the following: The mBot receives a pre-drawn finished house. It systematically traverses the house using the line sequence sensor. With each new attempt, it looks for a different path. According to Wikipedia (https://de.wikipedia.org/wiki/Haus_vom_Nikolaus), there are 44 valid and 10 invalid solutions from starting point 1.
   In extensive tables, the mBot remembers in particular the invalid solutions and at least one valid solution. This consists of 8 digits (= corner points) in the correct order. This would then be a further, somewhat advanced learning success.

Programme download: mBot2_draws_Nicolaus_House

YouTube video: https://youtu.be/8pYnir1flAY

12.2.4 Follow hand

still missing

12.2.5 Reaction test

still missing

12.2.6 Labyrinth - automatic

still missing

12.2.7 Labyrinth - manual

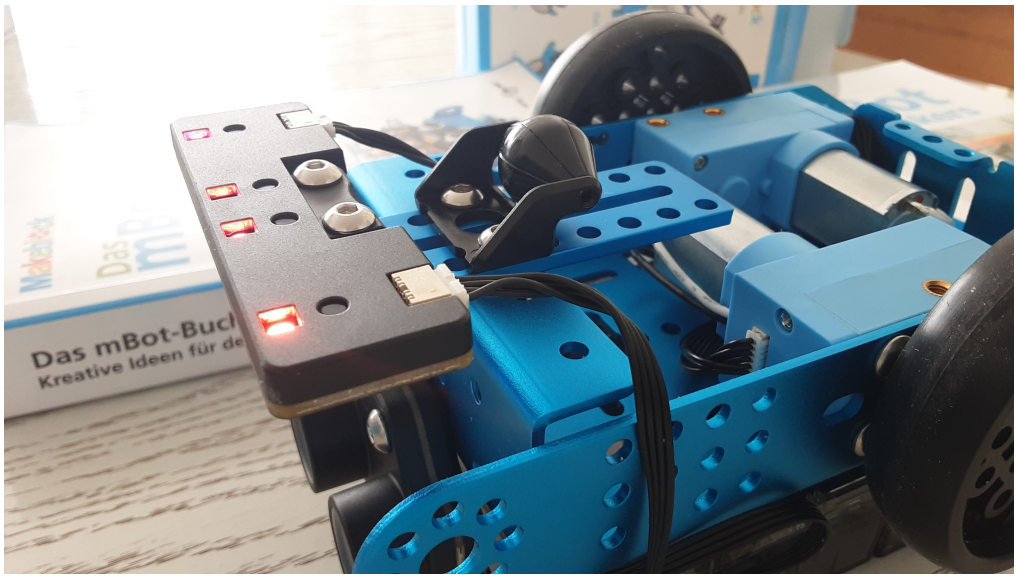
still missing

12.2.8 Crash avoidance

One of the most interesting applications for the mBot is fall prevention, e.g. on the edge of a table.

With the mBot 1, I had attached a second line-following sensor so that crash avoidance would work reliably even at an acute angle.

The mBot 2 has four line sensors right from the start, which are also supposed to work much more reliably. However, it didn't work straight away; the edge of the table was detected much too late, so I mounted the sensor about 3 cm further forward:

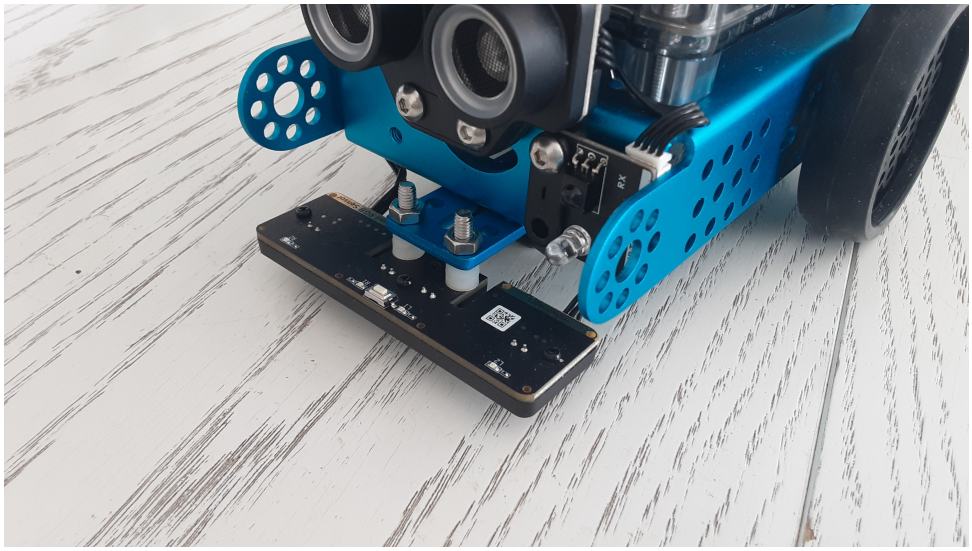


First of all, it was quite successful.

However, the table on which I am trying out the whole thing has a relatively pronounced grain. Because of this, a supposed edge was often detected and the mBot turned around in the middle of the table. I have not found a way to change the sensor sensitivity. This could also be because the help function is not yet available (10. 6. 2021).

After various offline experiments with a white index card, I then lowered the sensor an additional centimetre. It now works very reliably

.



However, there was one problem. Since the whole thing worked so well, I let the mBot move completely freely. I had not considered that one end of the table was very intensively illuminated by the sun. This obviously had a strong effect on the sensor and the mBot fell off the table. Fortunately, it survived completely unscathed.

Programme download: mBot2_Absturz_1.mblock (581.1 KB)

YouTube video: https://youtu.be/RNu612wHb2s

12.2.9. wolf-goat-kale

1. control via IR remote control

Actually, controlling the mBot 1 via the IR remote control is a bit confusing.

There, the objects are selected via the

Arrow left, 1, 4 and 7 resp.

Arrow right, 3, 6 and 9

controlled. I wanted to change that (also in anticipation of joystick control).

I have therefore no longer used eight keys but only the four arrow keys. These move the active object (printed in green) in the respective direction.

When selecting the object to be moved vertically, I did not want to stop at the upper or lower edge but instead jump to the lowest (uppermost) object. For this behaviour, the following formula is obvious for the downward movement: New = (Active mod 4) + 1. The mod function first determines the value zero for the new object, to which one is then added.

For the opposite direction, I thought for quite a long time until I came up with the function New = (Active + 2) mod 4 + 1, which seems strange at first glance: If Active is at the top position, i.e. has the value 1, the new value 4, i.e. the lowest position, is determined for the move up.

Here are the two blocks:

The block vertical is used to set the colour of the "old" object to white and the colour of the "new" object to green.

The movement of the object is no longer erratic but gradual.

Programme download: [mBot2_W_Z_K _IR.mblock(197.9 KB)]

2. control via joystick

It was during this type of input that I became aware of the mBot 2's new multi-tasking option. I was a little surprised that there is no direct polling option for the joystick and the Cyberpi's two buttons. Instead, separate blocks can and must be created for this, running in parallel. The main programme can also be started in several instances. I have realised this for the check for an invalid move and the check for the end of the programme.

Unfortunately, it is probably not possible to use list elements as sprites. Instead of a list with four elements for the four objects, I had to create four individual variables line1 - line4 instead. These have to be addressed individually: Visible especially in the *veritcal*, *horizontal* and *output* blocks. Using a list, these could be much more compact.

Programme download: mBot2_W_Z_K.mblock (195.5 KB)

YouTube video: https://youtu.be/YrAMQ4dbyo4

12.2.10. House of Father Christmas

Actually, I wanted to realise the operation of this game via the joystick so that the IR remote control does not have to be bought additionally.

However, this turned out to be very impractical: I had realised it in such a way that when I dragged the joystick upwards, the next clockwise corner point was activated, and when I dragged it downwards, the previous counterclockwise corner point was activated. When pressing the joystick, the connection between the last end point and the current one was drawn in.

This became very confusing at the moment when more than three of the eight possible connections already existed.

I therefore decided to implement the much clearer version, which is also available on the mBot 1: By pressing one of the buttons A - E on the remote control, the connection is drawn in.

The control of the screen on the mBot 2 differs significantly from that on the mBot 1.

In the latter case, with the extension TFT-LCD, any graphic elements (line, rectangle, circle) can be drawn on the screen and any text can be output.

With the mBot 2 it is different: If both text and graphic elements are to be displayed, the extensions "Figures" and "Scribble" must be used. This makes the handling much more complicated.

Programme download: mBot2_Nikolaus_IR_2.mbloc (245.1 KB)

Youtube video: https://youtu.be/pcv1-Zhlrpw

12.2.11. Towers of Hanoi

still missing

12.2.12. Tic-Tac-Toe

still missing

12.2.13. Conway's Game of Life

still missing

12.2.14. Nim games

still missing